Chapter 4

Spiking neural model

In the previous decade, the majority of artificial neural network models were based on a computational paradigm involving the propagation of continuous variables from one unit to another (Rumelhart and McClelland, 1996). However, in recent years experimental evidence has been accumulating to suggest that biological neural networks, which communicate through spikes, use the timing of these spikes to encode and compute information (Abeles, 1991; Thorpe et al., 1996; Rieke et al., 1997). Several alternative schemes to the rate–coding hypothesis have been proposed, suggesting that information may be coded in synchronization and oscillations of populations of neurons, or in the precise timing of the neural pulses (Recce, 1999; Gerstner and Kistler, 2002).

A new generation of *pulsed* neural networks has emerged, which focuses upon the mathematical formalization of the computational properties of biological neurons (Gerstner, 1995; Maass, 1997; Stevens and Zador, 1998). The neural models created capture the spiking nature of the neurons and retain the essentials of the behavior to be modeled, while trying to simplify the description (Gerstner, 1999; Izhikevich, 2001). This chapter focuses upon the description of a simplified spiking neural model and of the type of computations that it can account for.

In order to understand how a biologically inspired neural model behaves it is necessary to offer a brief introduction in the physiology of the real cell (Section 1). Neural activity of biological cells may be described at several levels of abstraction. In Section 2, the main tendencies in neural modeling, i.e., detailed vs. simplified, are reviewed, outlining the advantages and drawbacks of each direction. This description is followed by the presentation

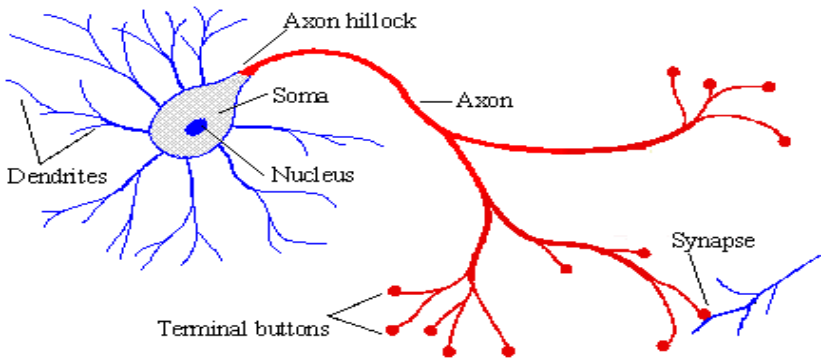


Figure 4.1: Schematic of a biological neuron. See in text for the description of the functionality of each component.

of a number of time-coding schemes and computation modes that are implemented in our simulator. In Section 3 the neural model implemented in SpikeNNS is described (e.g., equations, parameters, input patterns). The last section offers a brief review of a number of learning rules for spiking neurons.

4.1 Biological neuron physiology

The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell that uses biochemical reactions to receive, process and transmit information. The neural unit represents the basic information processing mechanism that is the foundation of human cognition.

4.1.1 Neural components

A typical neuron can be divided into three functionally distinct parts, namely the *dendrites*, the *soma*, and the *axon* (see Figure 4.1). A neuron receives connections from thousands other neurons. Most of these contacts take place on the neuron dendritic tree, however they can also exist on the soma or the axon of the neuron. The morphology of the dendritic tree plays an important role in the integration of the synaptic inputs and it influences the way the neuron processes information and computes (Mel, 1994). The strengths of the charges received by a neuron on its dendrites are added together through a nonlinear process of spatial and temporal summation (Koch, 1999). The resulting input flows to the soma and

suffers voltage attenuation, so that about half of the total charge injected into the distal dendritic site reaches the soma.

Two parameters play a critical role in the spatio–temporal integration of synaptic inputs: the *space constant* λ that gives the voltage attenuation with distance and the *membrane time constant* τ_m, which describes how fast the voltage decays. The input transmission from a dendritic site to the soma is characterized by a *propagation delay*. In the case of slowly varying inputs this is dictated by the membrane time constant value, but for fast synaptic inputs it outpaces τ_m, giving the possibility of submilisecond coincidence detection in dendrites (Koch, 1999).

The primary function of the soma is to perform the continuous maintenance required to keep the neuron functional (Kandel et al., 2000). The part of the soma that performs the important non–linear processing of the input is the *axon hillock*. If the total input produces a depolarization up to the neural *threshold* (i.e., of about -55 mV), the axon hillock fires an action potential. The output signal is transmitted down the axon, which delivers it to other neurons.

4.1.2 The action potential

The firing of a neuron, referred to as the *action potential*, is an all or none response. This means that, incoming stimuli either produce action potentials, if they exceed the neuron's threshold value, or they do not. A spike or action potential is a stereotyped impulse of fixed magnitude generated by the neuron. After the firing of an action potential, the neuron enters a *refractory period* when no further action potentials can be generated. Even with very strong input, it is impossible to excite a second spike during or immediately after a first one. This causes that action potentials in a spike train are usually well separated. The minimal distance between two spikes defines the *absolute refractory period* of the neuron. The absolute refractory period is followed by a phase of *relative refractoriness* where it is difficult, but not impossible, to generate an action potential (Kandel et al., 2000; Gerstner and Kistler, 2002).

4.1.3 Synapses

The site where the axon of a presynaptic neuron makes contact with the dendrite (or the soma) of a postsynaptic cell is the *synapse*. The most common type of synapse in the ver-

tebrate brain is a chemical synapse (Shepherd and Koch, 1990). When an action potential arrives at a synapse, it triggers a complex set of biochemical processes that lead to a release of neurotransmitters from the presynaptic terminal into the synaptic gap. The voltage response of the postsynaptic neuron to a presynaptic action potential is referred to as the *postsynaptic potential*. A single synaptic input is rarely sufficient to determine the generation of an action potential. This is usually triggered as a consequence of the nonlinear interaction of several excitatory and inhibitory synaptic inputs (Mel, 1994; Poirazi et al., in press). At the end of the synaptic action, the postsynaptic potential decays exponentially towards the resting value with a rate of decay given by the time constant τ_m. The physical and neurochemical characteristics of each synapse determine the strength and polarity of the new input signal. Synapses are believed to be the locus of learning and memory in the brain (Squire and Kandel, 1999). This is where the brain is the most flexible and the most vulnerable.

4.2 Computational and modeling aspects

The idea that brains are computational in nature has captured the attention of many researchers in the last decade (Churchland and Sejnowski, 1992; Maass, 1995; Koch, 1999; O'Reilly and Munakata, 2000). In a strict sense, a physical system computes a particular function if there is an appropriate mapping between the physical states of the systems and the elements of the function and if the mapping of the input space into the output space performed by the function can be described in terms of some rule (McCarthy, 1963; Churchland and Sejnowksi, 1992). For instance, one can describe the stochastic behavior of a neuron that maps an activation value V to a binary response that is a *spike* in 25% of the cases and *no spike* in 75% cases with a probability distribution function: $f : V- > (g : \{spike, no\, spike\}- > [0, 1])$ (after Touretzky, 2001).

The studies concerned with the way the brain represents and computes information based on the activity of neural components and using as main methodology the computer simulations, are unified under the computational neuroscience discipline (see the introduction to the field in Section 1.1). This research stream differs from other approaches of neural modeling in that it believes that understanding the way the brain computes is very closely dependent on the knowledge of the anatomical and physiological details of neural elements (Bower and Beeman, 1998). Hence, we can talk about a general tendency in computational

neuroscience towards realistic simulation of the structure and physiology of the biological neurons.

4.2.1 Levels of detail in neural modeling

Neural activity may be described at several levels of abstraction. We can distinguish between two major modeling directions in computational neuroscience, depending on how much of the neuron details the modeler wants to take into account. That is, because the more details are taken into account, the greater are the computational demands of the model.

Detailed neural models

One trend in computational neuroscience is oriented towards creating ever more detailed and complex models (De Schutter and Bower, 1994 a,b; Bower and Beeman, 1998; Segev and Burke, 1998; Steuber and De Schutter, in press). The detailed modeling of the neurons aims to describe the chemical processes undergone at the subcellular level, including the biophysics of ionic channels and dendritic trees, the synaptic interactions between excitation and inhibition, the voltage-dependent events in the active dendrites (Destexhe et al., 1998; Mainen and Sejnowski, 1998). In this effort, the neurobiologists argue that one has to consider the anatomical and physiological details of the neurons if he/she pursues a full understanding of the nervous system (Bower and Beeman, 1998).

The model proposed by Hodgkin and Huxley in 1952 offers perhaps the clearest instance of a detailed model of the processes involved in action potential generation. The model provides a detailed description of the biophysics of ionic mechanisms underlying the initiation and propagation of the neural spike. By doing this, it offers an accurate *quantitative* model of the physiological data. However, complex frameworks like the Hudgkin-Huxley model, which account for numerous ions channels and different types of synapses are difficult to construct and to analyze. An important conceptual drawback of this family of models is that their numerical complexity (e.g., solve a large number of nonlinear differential equations) can prevent one from understanding which features are responsible for a particular phenomenon and which are irrelevant (Koch, 1998).

Hodgkin-Huxley-like models can be applied to point neurons, that is, neurons without any spatial structure. However, the morphology and architecture of the dendritic tree plays an

important role on the nonlinear functioning of the neurons (Mel, 1994; Vetter et al., 2001). For the construction of detailed neuronal models that consider all of the cells complexities, including the branched cable structure, the standard approach is to divide the neuron into a finite number of isopotential connected compartments (Rall and Agmon-Snir, 1998; Segev and Burke, 1998). Each compartment acts like a capacitance-resistance circuit, modeled by a system of ordinary differential equations. It can represent somatic, dendritic or axonal membrane and contain a variety of synaptic inputs (Segev and Burke, 1998).

Compartmental models are implemented by powerful, biologically realistic simulators, such as Genesis (Bower and Beeman, 1998) and Neuron (Hines and Carnevale, 1998). Nevertheless, there is an important computational cost in memory and speed that detailed simulations based on compartmental models must pay. For instance, modeling of calcium dynamics in a *single* Purkinje cell, using 4550 compartments and 8021 ionic channels required 8 Sun workstations running one hour for the simulation of 550 ms of Purkinje cell activity (De Schutter and Bower, 1994a, b).

A major argument in favor of detailed modeling is that the close replication of the nervous system structure as a basis of exploring its functions increases the chances to discover its unknown (or unsuspected) organizational principles (Bower, 1997). The drawbacks arrive from the complexity of these models. That is, they are difficult to construct and analyze; they scale poorly with the network size and activity; they represent a quantitative, rather than a qualitative description of the neural behavior, which can prevent one from understanding the crucial features of the system. Despite the recent efforts oriented for the creation of simulation frameworks that allow modeling of large-scale networks of biologically realistic neurons (see Parallel GENESIS, Bower and Beeman, 1998; Goddard et al., 2001), the detailed models are generally considered more suitable for the study of single neurons or small networks behavior.

Formal spiking neuron models

The second stream of research in computational neuroscience is oriented towards modeling the spiking nature of the neurons and retaining the essential elements of the behavior being modeled, while trying to simplify the complexity of the resulting description (Gerstner, 1991; Maass, 1995; Maass, 1997; Rieke et al., 1997). The principal motivation for the creation of simplified models is that they allow studying more easily the computational and

functional principles of neural systems (Koch, 1999).

The reduction of the detailed neuron models to formal models requires simplifications in at least two respects. First, the non–linear dynamics of spike generation must be reduced to a single ordinary differential equation and second, the spatial structure of the neuron (i.e., the dendritic tree) is neglected and reduced to an input (Gerstner and Kistler, 2002). To support the validity of the former simplification, Kistler et al. (1997) demonstrated that spike generation in the Hodgkin–Huxley model can be reproduced to a high degree of accuracy (i.e., up to 90%) by a single variable model. The authors pointed out that the Hodgkin–Huxley model shows a sharp, threshold–like transition between an action potential for a strong stimulus and graded response (no spike) for slightly weaker stimuli. This suggests that the emission of an action potential can be described by a threshold process (see also Section 4.3.1).

Several simplified neural models have been proposed in the last decades. The *leaky integrate–and–fire* neuron is probably the best–known example of a formal neural model (Tuckwell, 1988; Bugmann, 1991; Stevens and Zador, 1998). It simulates the dynamics of the neuron membrane potential in response to a synaptic current by implementing an equivalent electrical circuit. The function of the integrate–and–fire circuit is to accumulate the input currents and, when the membrane potential reaches the threshold value, to generate a spike. Immediately after emitting a pulse, the potential is reset and maintained there for an absolute refractory period.

The simplified mathematical models for spiking neurons cannot account for the entire range of computational functions of the biological neuron. Rather, they try to abstract a number of essential computational aspects of the real cell function. The essential features implemented can differ between models, as a function of what the modeler considers to be relevant and crucial for its domain study. Thus, the *integrate–and–fire model* focuses upon the temporal summation function of the neuron (Bugmann and Taylor, 1997). The *spike response model* proposed by Gerstner (1999) simplifies the action potential generation to a threshold process. The *resonate–and–fire model* (Izhikevich, 2001) focuses upon the operation of the neuron in a resonating regime. By contrast with the detailed neural models, the computational strength of the spiking neurons arises from the way they interact with each other, when they work cooperatively in large networks.

In summary, this section presented a brief description of the main modeling directions in computational neuroscience. The aim of this was to outline the advantages and drawbacks

of each research stream. The review indicates that there is a trade–off problem that a researcher has to solve, when choosing the appropriate level of detail for the task to be simulated. Another aspect, which differentiates the two approaches, concerns the time scale at which they work. Detailed neural models are more appropriate to investigate neural patterns of activity that emerge within a short–time scale (i.e., hundred of ms of seconds). Conversely, networks of spiking neurons can be used to model learning in developmental processes (see Koch, 1999).

As stated in the introduction to this thesis, the general aim of our work is to investigate how cognitive phenomena such as visuomotor coordination, can emerge through development, from the properties of basic elements when they interact and function cooperatively. Consequently, our simulation work will be built using a simplified neural model, considered to be best suited to our modeling purposes. In the remainder of this section, we describe what types of computations with spiking neurons are accounted for in our model.

4.2.2 Neural communication with spikes

Neurons communicate by producing sequences of fixed size electrical impulses called action potentials or spikes (Adrian, 1926). As Rieke and colleagues puts it:

> Spike sequences are the language for which the brain is listening, the language the brain uses for its internal musings, and the language it speaks as it talks to the outside world (Rieke et al., 1997, page 1).

In the theory of neural information processing, there are two main hypotheses with respect to where in the spike train the neural information is encoded: in the neural firing rate or in the precise timing of the spikes. These hypotheses are introduced in turn, below.

Rate coding

Adrian (1926) introduced the concept of *rate coding*, by which the number of spikes in a fixed time window following the onset of a static stimulus code for the intensity of the stimulus. Since Adrian's studies, the rate coding hypothesis has been dominant in the neural computational field (see for a review Recce, 1999). The definition of the rate has been applied to the discovery of the properties of many types of neurons in the sensory, motor, and central

nervous system, by searching for those stimuli that make neurons fire maximally (Kandel et al., 2000).

Recent observations on the behavior of cortical visual neurons demonstrated a temporal precision in brain function that is higher than would be predicted from frequency coding (Abeles et al., 1993; Thorpe et al., 1996; Abeles and Gat, 2001). This suggests that firing rate alone cannot account for all of the encoding of information in spike trains. Consequently, in the last decade, the focus of attention in experimental and computational neuroscience has shifted towards the exploration of how the timing of single spikes is used by the nervous system (Gerstner and Kistler, 2002; see Section 4.2.2).

It is important to understand that the pulse coding represents an extension of the way neurons code information, rather than a replacement of the firing rate code. Panzeri and Schultz (2001) proposed such a unified approach to the study of temporal, correlation and rate coding. They suggest that a spike count coding phase exists for narrow time windows (i.e., shorter than the timescale of the stimulus-induced response fluctuations), while for time windows much longer than the stimulus characteristic timescale, there is additional timing information, leading to a temporal coding phase.

Temporal coding by relative latencies

In a temporal code, information can be contained in the *temporal pattern* of spikes (inter–spike interval codes) or in the *time–of–arrival* of the spike (relative spike timings) (Cariani, 1997). In the following, we discuss the later coding scheme, which is implemented in SpikeNNS.

Neurobiological studies of sensory coding of stimuli in the auditory and visual systems revealed that *latency of transmission* is a potential candidate for coding the stimulus features (Bugmann, 1991; Heil and Irvine, 1996; Heil, 1997). An example is the study by Gawne et al. (1996) who showed that the latency of neurons response in the striate cortex is a function of the stimulus contrast and that synchronization based on spike latencies can make an important contribution to binding contrast related information. The coding scheme which represents analog information through differences in the firing times of different neurons is referred to as *delay coding* or *latency coding* (Hopfield, 1995; Gawne et al., 1996; Maass, 1997; Thorpe and Gautrais, 1998).

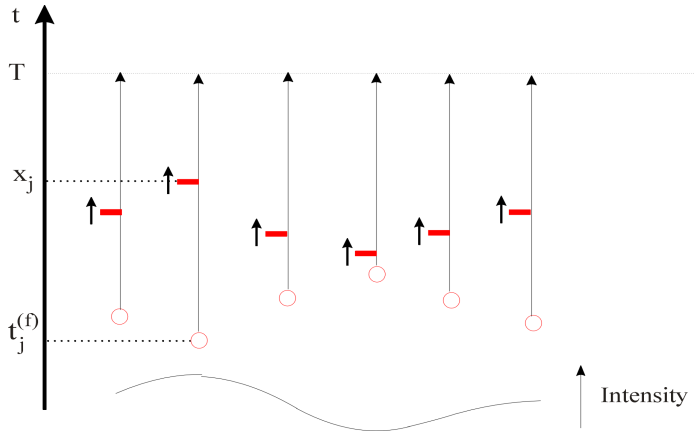


Figure 4.2: Coding by relative delay. The neurons in figure emit spikes at different moments in time $t_j^{(f)}$. The most strongly activated neuron fires first (i.e., second from left). Its spike travels a considerable distance along the axon, until last neuron fires (i.e., the fourth from left). The latencies x_j are computed with respect to a reference time T (adapted after Thorpe et al., 2001).

According to Hopfield (1995) and Maass (1997), a vector of real numbers $(x_1, ..., x_n)$ with $x_j \in [0, 1]$ can be encoded in the firing times t_j of n neurons, such as $t_j^{(f)} = T - c \cdot x_j$, where T is some reference time and $c \cdot x_j$ represent the transmission delays. The timing can be defined relatively to some other spike produced by the same neuron or to the onset of a stimulus. If for each neuron, we consider only the latency of the first spike after the stimulus onset, then we obtain a coding scheme based on the *time–to–first–spike*.

According to Van Rullen and Thorpe (2001) cells can act as 'analog–to–delay converters'. That is, the most strongly activated cells will tend to fire first and will signal a strong stimulation, whereas more weakly activated units will fire later and signal a weak stimulation. This coding scheme was proposed by Thorpe and Gautrais (1998), who argued that during visual object recognition the brain does not have time to evaluate more than one spike from each neuron per processing step. The idea is supported by other experimental studies (Tovee et al., 1993) and was used to implement learning in a number of neural network models, based on the timing or the order of single spike events (Ruff and Schmitt, 1998; Van Rullen et al., 1998, see also Section 6.1).

4.2.3 Computational properties of spiking neurons

Spiking neural models can account for different types of computations, ranging from linear temporal summation of inputs and coincidence detection to multiplexing, nonlinear operations and preferential resonance (Koch, 1999; Maass, 1999). Several recent studies employing rigorous mathematical tools have demonstrated that through the use of temporal coding, a pulsed neural network may gain more computational power than a traditional network (i.e., consisting of rate coding neurons) of comparable size (Maass and Schmitt, 1996; Maass, 1997).

A simple spiking neural model can carry out computations over the input spike trains under several different modes (Maass, 1999). Thus, spiking neurons compute when the input is encoded in temporal patterns, firing rates, firing rates and temporal corellations, and space–rate codes. An essential feature of the spiking neurons is that they can act as coincidence detectors for the incoming pulses, by detecting if they arrive in almost the same time (Abeles, 1982; Softky and Koch, 1993; Kempter et al., 1998).

When operating in the integration mode (see the integrate-and-fire model in Section 4.2.1), the output rate changes as a function of the mean input rate and is independent of the fine structure of input spike trains (Gerstner, 1999). By contrast, when the neuron is functioning as a coincidence detector, the output firing rate is higher if the spikes arrive simultaneously, as opposed to random spike arrival. More precisely, the neuron fires (e.g., signals a detection) if any two presynaptic neurons have fired in a temporal distance smaller than an arbitrary constant $c1$, and do not fire if all presynaptic neurons fire in a time interval larger than another constant $c2$ (Maass, 1999).

For a neuron to work as a coincidence detector, two constraints have to be satisfied: (1) the postsynaptic potential has to evolve in time according to an exponential decay function and (2) the transmission delays must have similar values, so that the simultaneous arrival of the postsynaptic potentials which cause the neuron to fire will reflect the coincidence of presynaptic spikes (Maass, 1999). Note that not any spiking neural model can detect coincidence. For instance, the resonator neuron fires if the input train of spikes has the same phase with its own oscillation, but has low chances to spike if the inputs arrive coincidentally (Izhikevich, 2000).

In SpikeNNS, neurons can compute in two regimes: coincidence detection and threshold–and–fire. Acting as coincidence detectors is more likely for hidden units, when they com-

pute over pulses coming from the input layers. That is, because in our implementation, the input spikes arriving on the afferent connections are affected by similar delays with a small noise factor. The latency of the spikes is given by the firing times of the input nodes. The operation in this computing domain depends also on the neural threshold and on the value of the membrane time constant that describes how fast decays the posystynaptic potentials (see Section 4.3.2).

In the threshold–and–fire mode, neurons perform a linear summation of the inputs in a similar manner with the integrate–and–fire model. The integration of pulses over a larger time interval is particularly required in the case of spikes arriving on the lateral synapses, which are affected by a large range of delays (e.g., from 1 to 10 ms). The shift between computing modes, i.e., coincidence vs. integration, and between different ways of integrating the input signals is illustrated by our simulations in Sections 6.1 and 6.2.1 and discussed in Section 7.1.3.

4.3 Neural model in SpikeNNS

The neural model implemented in SpikeNNS is a simplified version of the Spike Response Model (Gerstner, 1991; Gerstner et al., 1993; Gerstner, 1999) referred to as SRM_0. The Spike Response Model (SRM) represents an alternative formulation to the well–known integrate–and–fire model. Instead of defining the evolution of the neuron membrane potential by a differential equation, SRM uses a kernel-based method. By doing this, the Spike Response Model is slightly more general than the integrate–and–fire models because the response kernels can be chosen arbitrarily, whereas for the integrate–and–fire model they are fixed (Gerstner, 1999). According to Kistler et al. (1997) the Spike Response Model can reproduce correctly up to 90% of the spike times of the Hodgkin-Huxley model. The model can also be used to simulate the dynamics of linear dendritic trees, as well as non–linear effects at the synapses (Gerstner, 1999). The Spike Response Model offers us a powerful computational framework that captures the essential effects during spiking and has the advantages of a simple and elegant mathematical formalization.

4.3.1 Spike Response Model

The Spike Response Model describes the state of a neuron by a single variable, the membrane potential V_i. Figure 4.3a shows the time evolution of the membrane potential of neuron i as a function of time t. Before any input spike has arrived at the postsynaptic neuron i, the variable $V_i(t)$ has the value 0. The firing of a presynaptic neuron j at time $t_j^{(f)}$ evokes a postsynaptic potential in the neuron i modeled by the kernel response ϵ_{ij}. Each incoming spike will perturb the value of V_i and if, after the summation of the inputs, the membrane potential V_i reaches the threshold θ then an output spike is generated. The firing time is given by the condition $V_i(t_i^{(f)}) = \theta$. After the neuron has fired the membrane potential returns to a low value which is described by the refractory period function η. After firing, the evolution of V_i is given by the equation:

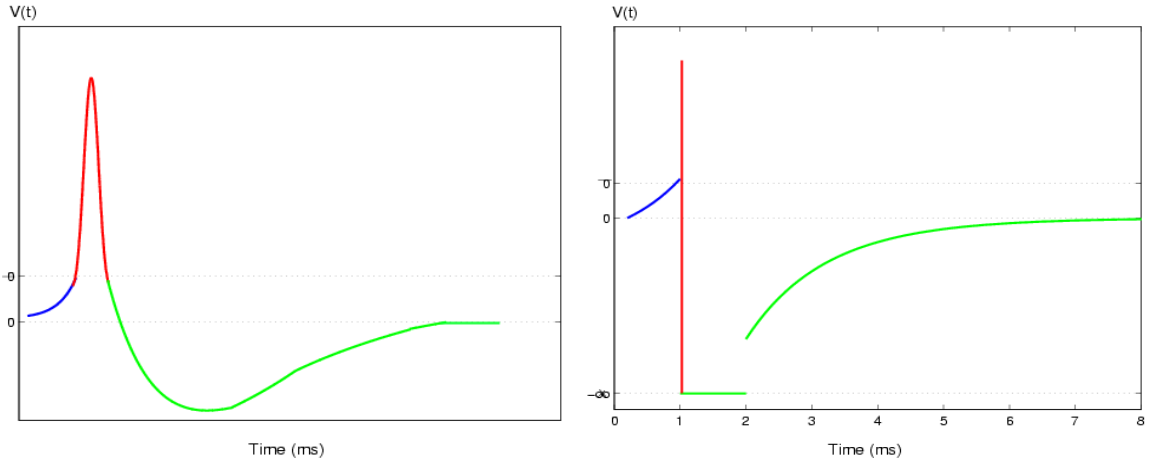
$$V_i(t) = \eta_i(t - t_i) + \sum_{j \in \Gamma_i} w_{ij} \sum_{t_j^{(f)} \in F_j} \epsilon_{ij}(t - t_i, t - t_j^{(f)}) + \int_0^\infty \tilde{\epsilon}(t - t_i, s)\mathcal{I}^{ext}(t - s)ds, \quad (4.1)$$

with $s = t - t_j^{(f)}$. The first term in the equation (i.e., the kernel η_i) accounts for refractoriness in the neuron behavior. The second term represents the contribution of all previous spikes $t_j^{(f)}$ of presynaptic neurons j on the membrane potential of neuron i. Γ_i denotes the set of neurons presynaptic to i, F_j is the set of all firing times of neuron j and w_{ij} are the synaptic strengths between cells (see Figure 4.5).

The kernel ϵ_{ij}, as a function of $t - t_j^{(f)}$, represents the time course of the postsynaptic potential evoked by the firing of the presynaptic neuron j at time $t_j^{(f)}$ (see Figure 4.4a). The evolution of the postsynaptic potential function depends also on the time $t - t_i$ that has passed since the last spike of the postsynaptic neuron. That is, because if the neuron is in a refractory period, its response to an input spike is smaller than if the neuron is fully responsive. The last term represents the effect on the neuron of an external driving current $\mathcal{I}^{ext}$ and the kernel $\tilde{\epsilon}(t - t_i, s)$ is the linear response of the membrane potential to the input current and depends on the time that has passed since the last output spike was emitted at t_i (Gerstner, 1999).

$\mathbf{SRM_0}$

A simpler version of the Spike Response Model can be obtained by neglecting the dependence of the ϵ_{ij} kernel on the term $t - t_i$ (i.e., the effect of the neuron's last spike on the



(a) Membrane potential in SRM

(b) Membrane potential in SpikeNNS

Figure 4.3: (a) Spike Response Model, the membrane potential V of neuron i as a function of time t. (b) SpikeNNS neural model, the time course of the membrane potential V_i. Θ is the neural threshold. See in text for more details on the time evolution of the membrane potential.

postsynaptic potential function) and by considering a null external current. Consequently, equation 4.1 can be rewritten:

$$V_i(t) = \eta_i(t - t_i) + \sum_{j \in \Gamma_i} w_{ij} \sum_{t_j^{(f)} \in F_j} \epsilon_0(t - t_j^{(f)}). \tag{4.2}$$

This version of the Spike Response Model has been entitled SRM_0 and has been applied for the analysis of computations with spiking neurons by Maass (1999).

The neural model implemented in SpikeNNS is completely specified by the set of Equations 4.2, 4.3, 4.5, which account for several important aspects of neural behavior: the spiking nature of the neuron, the attenuation of the response at the soma resulting from synaptic input, the absolute and relative refractory periods. The model also accounts for spike latency and noise in the neural response (see description in sections below).

Figure 4.3b shows the time evolution of the membrane potential V_i in the simplified neural model implemented in SpikeNNS. Compared with the membrane potential in the SRM model represented in Figure 4.3a, the shape of the action potential is reduced to a formal event, captured by a δ pulse (the vertical line). After the spike emission, the membrane voltage is reset to a negative value and is kept there for 1 ms (see Section 4.3.3). The ascending

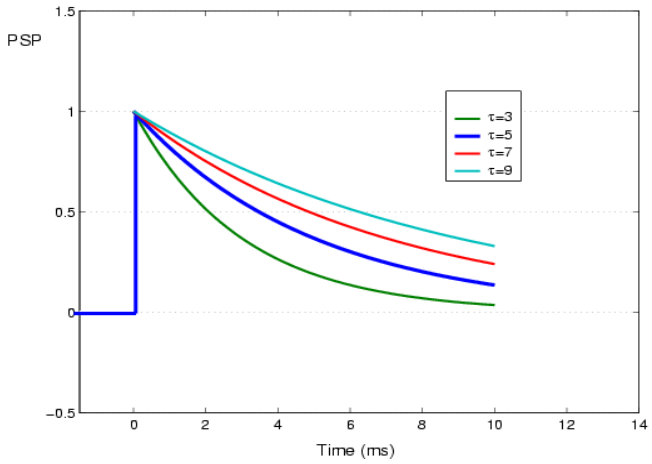


Figure 4.4: Postsynaptic potential function in SpikeNNS. The curve decay with time is plotted for different values of the decay rate, given by the membrane time constant τ_m. Note that the slope of the curve is negatively correlated with the value of the membrane time constant.

curve of the ϵ function also reduces to a pulse, followed by an exponential decay of the postsynaptic potential (see Section 4.3.2 below). In SpikeNNS, Equation 4.2 is implemented by the activation function ACT_Spike (Section 5.1), which expresses how the membrane potential V of a node i is calculated at a given time t.

Let us now consider the mathematical description of the two kernels ϵ and η required in the Equation 4.2.

4.3.2 Postsynaptic potential function

In SpikeNNS, each hidden neuron is connected to a number of other neurons either from the same layer or from an input or another hidden layer. The firing of any node, i.e., input or hidden, is transmitted to all its postsynaptic units, where it evokes a postsynaptic potential of some standard form (see Figure 4.4). The spike transmission is affected by a noisy delay d, which in our implementation is proportional with the Euclidian distance between the presynaptic and the postsynaptic node (see Section 5.3.2). This delay corresponds to the axonal and dendritic transmission delay of real neurons (Koch, 1999).

When the presynaptic spike reaches the postsynaptic unit, the postsynaptic potential (PSP) jumps to a maximum value, i.e., in our simulation this value is set to 1. Afterwards, it decays exponentially towards the resting value, with a rate being given by the time constant

τ_m. In our model, the postsynaptic potential ϵ_{ij} is described as a function of the difference $s = t - t_j^{(f)} - d$:

$$\epsilon_{ij}(s) = exp(-\frac{s}{\tau_m})\,\mathcal{H}(s) \tag{4.3}$$

with

$$\mathcal{H}(s) = \begin{cases} 1, \text{ if } s \geq 0, \\ 0, \text{ otherwise,} \end{cases} \tag{4.4}$$

where t is the time of consideration, $t_j^{(f)}$ is the time of the presynaptic node firing and d is the delay on the connection. The Heaviside function sets the postsynaptic potential to a null value, for any time moment t that precedes the arrival of the presynaptic spike, that is $t < t_j^{(f)} + d$. Note that in our implementation the postsynaptic potential can only take positive values. Negative values of the postsynaptic potential, corresponding to the inhibitory synapses, are obtained through the multiplication of the PSP value by a negative weight value.

The decay of the postsynaptic potential with a rate given by the membrane time constant τ_m reflects the attenuation with time of the synaptic inputs in the biological neuron (see Section 4.1). The choice of this parameter has a significant effect on the way multiple synaptic inputs are integrated in time (see below).

Temporal summation of postsynaptic potentials

A single synaptic input is rarely sufficient to generate an action potential. The response of a neural cell is usually determined by the way it integrates multiple synaptic inputs. The basic arithmetic that dendrites of a real neuron compute is still a matter of controversy (Poirazi and Mel, 2000). Both linear and nonlinear interactions between synaptic inputs in the brain have been described by neurophysiological experiments (Cash and Yuste, 1999; Koch, 1999) and explored computationally with different formalisms (Rall, 1977; Mel, 1992). In SpikeNNS, we consider that both excitatory and inhibitory inputs accumulate linearly in time.

The total synaptic input to a hidden neuron i at some moment t is given by the contribution of all previous spikes of the presynaptic neurons (see Figure 4.5). The set of presynaptic neurons to the node i is $\Gamma_i = \{j \mid j \text{ is presynaptic to } i\}$. The set of all firing times of the

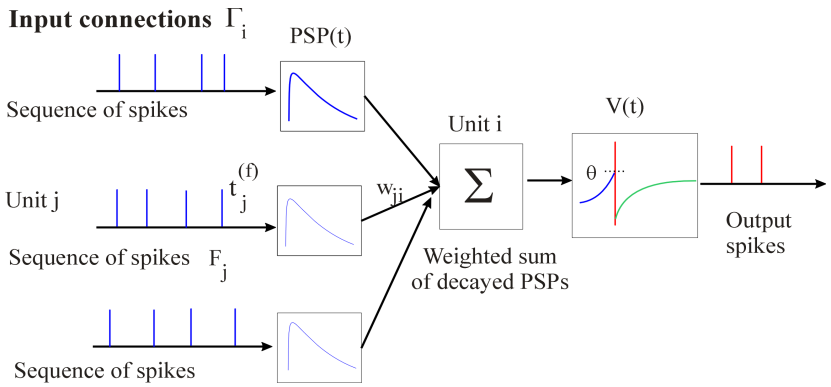


Figure 4.5: The presynaptic contribution to the membrane voltage $V(t)$ of neuron i. Each presynaptic neuron j in Γ_i emits a series of spikes F_j with firing times $t_j^{(f)}$. Neuron i computes the weighted sum of the decayed PSPs. If the sum exceeds the threshold then an output spike is generated.

presynaptic node j is given by $F_j = \{\, t_j^{(f)} \,|\, V_j(t_j^{(f)}) = \Theta\}$. In SpikeNNS, a limited number of spikes per neuron are stored (e.g., a maximum of 10 spikes/neuron were stored for the simulation run in Section 6.1).

In our model, the slope of the postsynaptic potential curve is negatively correlated with the value of the membrane time constant τ_m (see Figure 4.4a). That is, for large values of τ_m, the postsynaptic potential persists longer and it allows the temporal summation of inputs that produce in this way, an aggregate PSP larger than would be elicited by an individual input. The neural threshold Θ and the membrane time constant τ_m represent the principal parameters in determining how many excitatory inputs are needed for a neuron to fire. For example, the choice of a membrane time constant $\tau_m = 5$ (the blue graph in Figure 4.4a) causes an exponential decay of the postsynaptic potential from the maximum value to 0, in about 10 ms. This relatively slow decay of the postsynaptic potential curve favors the significant summation of the synaptic inputs which arrive in a time window no larger than $4 - 5$ ms. For instance, given the above value of the τ_m, in the simulations run in Chapter 6, the threshold values were set so that, at least three synaptic inputs (e.g, most commonly 4 or 5 inputs) were necessary for a postsynaptic spike to be emitted.

4.3.3 Refractoriness

After emitting the spike a node enters a refractory period. In the SRM_0 model the neuron behavior in the refractory period depends only on the last firing moment t_i. This means

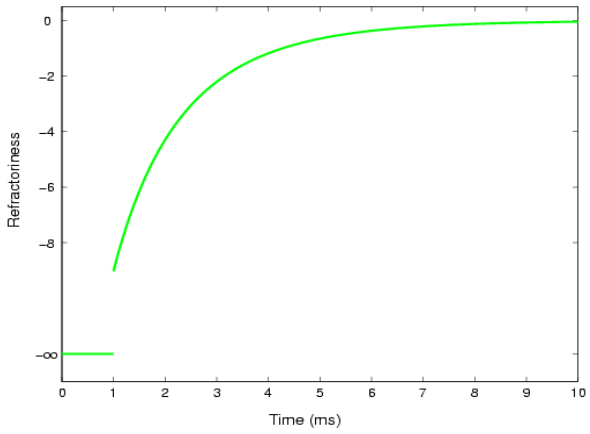


Figure 4.6: Refractory function in SpikeNNS. The Parameters $m = 0.8$ and $n = 3$. See in text for the meaning of these values.

that the effects of any previous own spikes are discarded. Gerstner (1999) referred to this as the 'short-term memory' approximation, that is, the neuron remembers only its most recent firing. The approximation can be considered correct if the intervals between two spikes are much longer than τ_m. This would make the influence of previous own spikes on the current refractory behavior, negligible. The refractory period is described as a function of the difference $u = t - t_i$ and is modeled by the equation:

$$\eta_i(u) = -\, exp[-u^m + n]\, \mathcal{H}'(u)\, \Theta. \tag{4.5}$$

with

$$\mathcal{H}'(u) = \begin{cases} \infty, & \text{if } 0 \leq u < 1, \\ 1, & \text{otherwise} \end{cases} \tag{4.6}$$

where Θ is the neural threshold and m and n are arbitrary parameters which adjust the slope of the decay curve (see Figure 4.6) (formula adapted after Sougné, 1999).

After a spike emission, the biological neuron behavior is characterised by the existence of two periods. First, there is an absolute refractory stage, while no further action potentials can fire. This is followed by a relative refractory period when it is difficult, but not impossible to generate an action potential (Section 4.1). In the neural model described here, the absolute refractory period lasts 1 ms, during which the Heaviside function (Equation 4.6) maintains the value of the η kernel at $-\infty$ (see Figure 4.6). The duration of the relative refractory period depends on the values of the parameters m and n. In Figure 4.6 these value are set so that the average inter–spike value is 10 ms.

4.3.4 Coding of the input patterns

When computing with temporal patterns in a network of spiking neurons, the input and output to the model is represented by vectors of *time series*, instead of vectors of analogical numbers, as in conventional neural network models (Maass, 1999). In our implementation, an input pattern consists of a series of firing times F_j of the input units j that encode the training values, as explained below.

The latency code described in Section 4.2.2 represents analog information through differences in the firing times of the neurons. Here, we define a coding scheme, where the analog values x_j can be encoded by the time advances of the input spikes to a reference time T_{int} (see Hopfield's formulation in Section 4.2.2). Figure 4.7 shows an input pattern represented by the spikes on five input units. An input spike has a large contribution on a postsynaptic node potential if it has just reached the node and a smaller effect if its arrival is less recent. Given equal transmission delays d, a vector of real values $(x_1, ..., x_n)$ with $x_j \in [0, 1]$ can be encoded in the postsynaptic potentials of the input spikes as a function of the difference between an arbitrary value T_{int} and the input units firing times $t_j^{(f)}$:

$$x_j = \epsilon(T_{int} - t_j^{(f)} - d) = exp\,(-\frac{T_{int} - t_j^{(f)} - d)}{\tau_m}). \tag{4.7}$$

Let us explain how a network can be trained with input values coded by this scheme. For instance, if the firing time of an input unit is $t_j^{(f)} = 2$ ms, the delay value is set to $d = 2$ms, $T_{int} = 10$ ms, and $\tau_m = 5$ ms, then a hidden unit receives a signal $x_j = 0.4$, given by the postsynaptic potential in Equation 4.7 (see first PSP value in Figure 4.7a). This value can be further used for the training of the hidden layer weights. For instance, in a self-organization process, weights are adapted as a difference between the input ($\epsilon = 0.4$) and the current value of the weight. Accordingly, the weights of a winner for this input will learn the value of 0.4. Note that, learning on the synapses of spiking neurons depends also on their output values, in a different way that it is the case for the continuous, rate–coding neurons. That is, learning is applied only to the excitatory synapses of those neurons, which fire. The later a hidden neuron fires, the smaller is the contribution of the input values on its afferent synapses change (see also Section 5.2.3).

In SpikeNNS, two coding schemes are used: *time coding* and *coding by synchrony*. Which scheme is used depends on the way the input patterns are specified and on the operation mode of the neurons (see Section 4.2.3). That is, neurons can either operate as integrators,

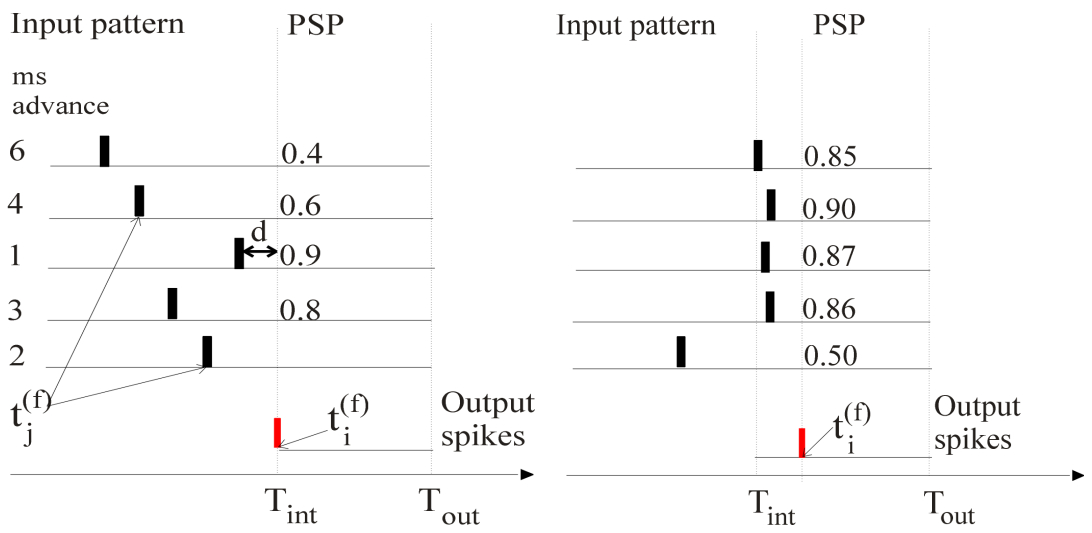


Figure 4.7: An input pattern consists of a train of spikes emitted at the times $t_j^{(f)}$. (a) A time coding scheme, where values are encoded in the spikes relative latencies to the T_{int} moment. In left the time advances of the input spikes are shown, computed as $T_{int} - t_j^{(f)} - d$. The PSP values represent the real values seen by the hidden units and which are used in training the synaptic weights. (b) Coding by synchrony, where the hidden neurons spike by detecting the coincidence in the firing of the input units.

by summing up the input values that arrive in a large time window, or they can function as coincidence detectors, by detecting the synchrony in the input spikes emission. Accordingly, the temporal coded inputs can be specified in two ways.

Firstly, real values can be encoded as described above, by using the spike time advances to a fixed reference time T_{int}. This method offers complete control over what values are encoded and received by the hidden units. Secondly, information can be encoded in the synchronous emission of the input spikes. This means that the hidden units can detect the coincident firing of a subset of input units. It represents a means to generate hidden units selectivity for a neural event that is characterised by the correlated firing of a set of presynaptic units.

With regard to how the time–coded input patterns are effectively implemented and applied in SpikeNNS, it consists of the following scenario. The series of firing times F_j corresponding to one pattern read from the input file, are converted in series of $1'$s and $0'$s that are applied to the hidden network at the times $t_j^{(f)}$. All the 1 values representing input spikes with the same timing $t_j^{(f)}$ are collected in a *subpattern* with that time stamp. By doing this, a single time–coded pattern generates as many subpatterns as different firing times are in the

original input pattern.

4.4 Learning with spiking neurons

Part of the work carried out on this thesis has focused upon the implementation of an extension of the SNNS simulator (Stuttgart Neural Networks Simulator) to support simulations with spiking neurons (see Chapter 5). The new simulator version, SpikeNNS, implements a general training framework adapted to the characteristics of learning with spiking neurons and temporal coded input patterns (see Section 5.2.3). By creating a general learning frame, the simulator is not limited to the learning rules currently existent in SpikeNNS, but it can be easily extended with any learning functions.

The current version of the simulator implements several learning rules for a self–organization process with spiking neurons. Issues of learning with spiking self–organizing maps have been discussed in Section 2.2.5. The adaptation rules implemented in SpikeNNS for the self-organization of a spiking neural map with plastic lateral synapses are described in Sections 6.1 and 6.2 along with the presentation of the simulations performed.

In Section 2.2.5, we emphasized the crucial role played by the horizontal connectivity in development in the real and artificial networks. Accordingly, significant modeling efforts have been dedicated to the integration of learning in lateral synapses in the models of cortical functions. Most of these attempts focused on exploring Hebbian–like association rules (see Section 2.2.5). More recently, experimental investigations shown that the relative timing of the pre– and post–synaptic neurons plays an important role in determining whether a synapse is potentiated or depressed (Markram et al., 1997; Zhang et al., 1998). Newer formulations of the Hebbian rule take into account the temporal order of spiking, so that potentiation occurs only if the postsynaptic excitatory potential precedes the firing of the postsynaptic neuron by at most 20 ms. Depression is the rule for the reverse order (Song et al., 2000). In the following, we describe briefly a number of learning rules that illustrate different ways to compute and learn with spiking neurons. Some of them will be implemented in future work.

Learning on neural firing rates. Choe and Miikkulainen (2000) proposed an adaptation of the lateral weights according to the Hebbian rule and based on the spiking rates of leaky

integrate-and-fire neurons:

$$w_{ij}(t) = \frac{w_{ij}(t-1) + \alpha \cdot V_i X_j}{\mathcal{N}}, \tag{4.8}$$

where $w_{ij}(t)$ is the connection weight between neurons i, j, $w_{ij}(t-1)$ is the previous value of the weight, α is the learning rate. V_i and X_j are the average spiking rates of the neurons computed after the network has reached a stable rate of firing. $\mathcal{N}$ is a normalization factor that prevents the excessive increasing of the weights. This learning algorithm was used successfully for image segmentation and contour binding by the synchronization of neural group activity (Choe and Miikkulainen, 2000).

Learning on the spike times synchronization. Sougné (1999) proposed a neurocomputational model for binding and inference that solves the binding problem by means of oscillation synchrony. Learning of the connection weight between two nodes i and j is defined by Equation 4.9. This expresses the fact that with every synchronous firing of the nodes, the weights w_{ij} as well as w_{ji} are increased by a constant value c:

$$\Delta w_{ij}^{+} = \begin{cases} c, \text{ if } t_j^{(f)} = t_i^{(f)} \\ 0, \text{ otherwise.} \end{cases} \tag{4.9}$$

The delays of the connections also undergo learning, so that at each synchronous firing of the nodes i and j, the delay d_{ij} as well as d_{ji} are set to a fixed value, which favors the synchronization of neurons at a certain frequency of oscillation. A decrease of the connection weights (i.e., forgetting of the binding) occurs if the two nodes do not fire inside a certain time cycle (i.e., up to 250 ms). This type of learning is successfully used to implement a short time memory, binding on multiple instantiation and with multiple predicates.

Spike–timing dependent plasticity (STDP). Song et al. (2000) proposed an adaptation of the Hebbian learning rule, which takes into account the relative timing of the pre- and postsynaptic action potentials. The synaptic modification arising from a pair of pre- and postsynaptic spikes separated by a time interval Δt is given by the function $F(\Delta t)$:

$$F(\Delta t) = \begin{cases} A_+ exp(\Delta t/\tau_+), \text{ if } \Delta t < 0 \\ -A_- exp(-\Delta t/\tau_-), \text{ if } \Delta t > 0 \end{cases} \tag{4.10}$$

where A_+ and A_- determine the maximum amounts of synaptic modification and the parameters τ_+ and τ_- determine the ranges of pre–to–post synaptic interspike intervals over which synaptic change occur. Strengthening of a synapse occurs for pre–synaptic action potentials which precedes postsynaptic firing by no more than 50 ms. Weakening of the

synapse is produced by a presynaptic action potential that follows the postsynaptic spike. The main role of this form of STDP is to strongly weaken causally ineffective inputs, hence, synaptic weakening dominates over synaptic strengthening.

Spike–timing learning has been mostly implemented for excitatory synapses, due to the rich experimental data showing how potentiation and depression of these synapses occurs in the brain. Conversely, there is much less neurobiological data from which to conceptualize the modification rules for inhibitory synapses, and consequently fewer cognitive models account for the plasticity of inhibitory synapses (see Roberts, 2000; Soto_Trevino et al., 2001).

Learning in the inhibitory synapses. In SpikeNNS learning in the inhibitory synapses is implemented according to the rule proposed by Levy and Desmond (1985). The authors proposed that presynaptic activity paired with postsynaptic inactivity leads to the potentiation of the active inhibitory synapse, while postsynaptic activity is required (indifferent of the presynaptic activity) for a loss of strength of the inhibitory synapse (see also Section 6.1). More recently, experiments in the mormyrid electric fish have suggested that pairing delays between pre- and post-synaptic spikes that cause excitatory synapses to decrease cause the inhibitory synapses to increase. Thus, if the postsynaptic inhibitory potential follows the postsynaptic spike within a time window of about 50 ms, than the inhibitory synapses is potentiated (Roberts, 2000).