Università degli Studi di Napoli Federico II

Dipartimento di Matematica e Applicazioni "Renato Caccioppoli"

Ph.D. Thesis

in

Scienze Computazionali e Informatiche - XXVI Ciclo

Analysis, tuning and implementation of neuronal models simulating Hippocampus dynamics

Author:
Dr. Pasquale De Michele

Supervisor:
Prof. Salvatore Cuomo

Tutor:
Prof. Eleonora Messina

Ph.D. Director:
Prof. Gioconda Moscariello

2014-2015

Declaration of Authorship

I, Dr. Pasquale De Michele, declare that this thesis titled, 'Analysis, tuning and implementation of neuronal models simulating Hippocampus dynamics' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Two things fill the mind with ever-increasing wonder and awe, the more often and the more intensely the mind of thought is drawn to them: the starry heavens above me and the moral law within me."

Immanuel Kant, *Critique of Practical Reason.*

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Dipartimento di Matematica e Applicazioni "Renato Caccioppoli"

Abstract

Scienze Computazionali e Informatiche - XXVI Ciclo

Doctor of Philosophy

Analysis, tuning and implementation of neuronal models simulating Hippocampus dynamics

by Dr. Pasquale De Michele

This work focuses on methods, algorithms and software for computational neuroscience, a research field where several computing strategies are intensively adopted for simulating real biological scenarios. This topic has a strong intersection with the Computer Science to build up frameworks that reproduce neuronal behaviours, coming from laboratory experiments and experiences. A key role is played by the simulation of synaptic mechanisms and neuronal dynamics that regulate the activity of the neurons. The model formalization requires long and deep steps in order to properly tune, through numerical simulations, a large number of biological parameters. These tasks, needed to validate a computational model, represent a well-known critical issue in terms of computing resources, both in time and in memory allocation.

The main target of this PhD Thesis is the study of Hippocampus brain region, devoted to the acquisition of new memory (i.e., storage) and retrieval of previously acquired (i.e., recall). The dissertation starts with an introduction to the biological context and with the state of the art on some single cell and network models. Then, the formalization and the implementation of computational schemes that reproduce typical neuronal phenomena are deeply investigated. More in detail, the depolarization block of a CA1 pyramidal neuron and the effects of the CREB protein activity increasing in a CA1 microcircuit are investigated. For these topics, sequential and parallel code packages, published on the well-known neuroscience repository *ModelDB*, are implemented. Finally, as a case of study the acquired know-how on the modelling of cell and network dynamics was adopted for reproducing user behaviours in a cultural heritage scenario. Here, the main idea is to measure the interest of an artwork spectator by means of models able to capture the context and the visitor behaviours.

The Thesis is structured as follows. **Chapter** 1 describes the preliminary notions and the state of the art on neuronal models. **Chapter** 2 reports the analysis, the formalization and the implementation of the computational model described in [6] (and published in [7]), while **Chapter** 3 is devoted to the same considerations on the computational model described in [8] (and published in [9]). Finally, **Chapter** 4 presents two biologically inspired computational models describing the personalized interactions of users with cultural heritage objects.

Acknowledgements

I wish to express my sincere thanks to all those who have guided, inspired and supported me: the list is quite long, but I hope I will not forget anyone.

Foremost, I would like to express my sincere gratitude to my advisor Prof. Salvatore Cuomo for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I wish to express my sincere thanks to my tutor, Prof. Eleonora Messina, for helping me during my Ph.D. experiences.

I would like to thank those who have always believed in me, in my work, and in my academic career, assisting me, with love, in every moment with words of comfort and reassurance.

I wish to express my sincere thanks to my colleagues who have been my travelling companions and with whom I have shared many experiences and emotions. The list is very long but I wish to address a particular thank to Giuseppe.

Thanks to my friends, in the course of this long period of study and work were close to me, always believing in the realization of this objective.

Finally, thanks to my family, for having made the greatest effort and for supporting me in every moment of my study career.

Contents

List of Figures

List of Tables

To those who have always believed in me

Chapter 1

Introduction to computational neurobiology

1.1 Structure of biological neurons

The nervous system is the site of the faculties of sense and thought and provides for the control of bodily functions: for this purpose, this system collects sensory information from the whole body and transmits them to the encephalon (i.e., brain) and spinal cord. The nervous system is divided in two principal sections, the central and peripheral nervous system. In particular, the central nervous system is composed by the brain and the spinal cord [66]. The first one is the main site of the integrative activity of the nervous system: in this, the memories are stored. The second one serves as a conductor for many nerve pathways to and from the brain, as well as for coordinating many not conscious nerve activities. More in detail, the human central nervous system involves a complex large-scale interconnected neural network consisting of approximately 10^{11} neurons [52].

A neuron (Fig. 1.1) is a cell with particular structural and functional features. It is characterized by a highly polarized structure able to facilitate the reception, integration and transmission of nerve impulses. As well as any other cell of human body, the material of which the neurons are constituted (i.e., *cytoplasm*) is surrounded by a thin "shell" called *cell membrane*, which separates these from the external environment.

The central part of the neuron is constituted by the *soma*, (i.e., the *cell body*) in which the *nucleus* resides. The soma ensures the necessary nourishment to the survival of the entire cell. Moreover, two types of cytoplasmic extensions, which confer to the neuron excitability and conductivity properties, arise from the soma. The first ones are the

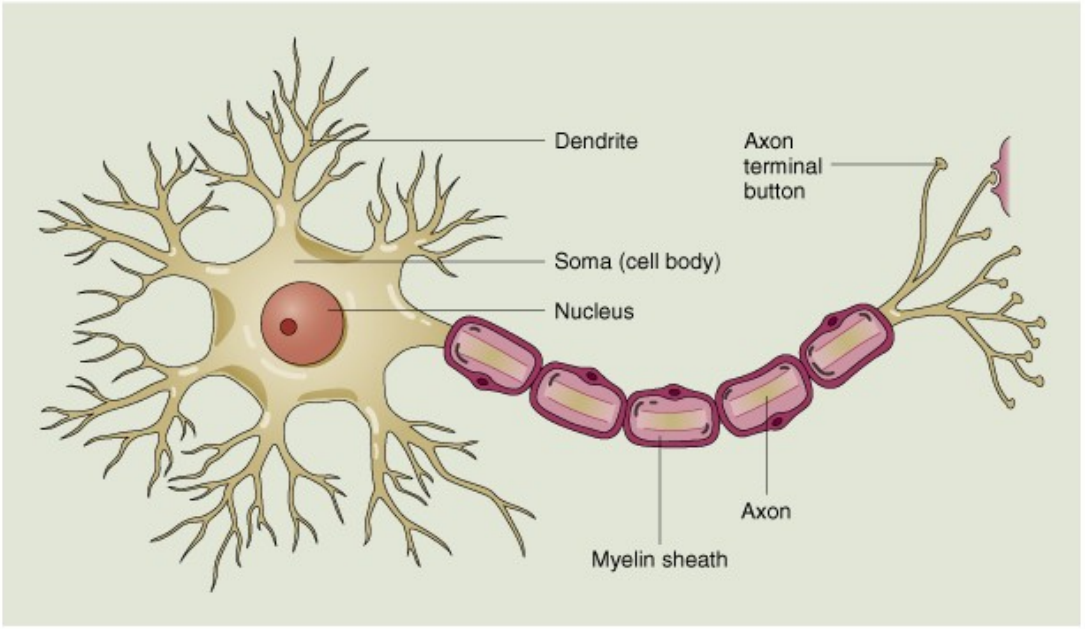

FIGURE 1.1: Schematic representation of a neuron.

dendrites, which are extensions to multiple branching that carry the nerve signal in a centripetal direction (i.e., toward the soma) and are the main receptive structures of the neuron (input). The second one is the *axon*, which transports the information generated from the soma to the dendrites of other neurons (output).

The distal part of the axon is divided into several ramifications, each of which ends in the *pre-synaptic terminal* (or *axon terminal bouton*, or *synaptic bouton*), which is an expansion that rests directly on the membrane surface of a dendrite or soma belonging to another neuron. The point at which the contact between a pre-synaptic terminal and the membrane occurs is called *electrical synapse* (Fig. 1.2 [91]).

When there is not a direct connection between the synaptic bouton and the dendrite or the soma, the synapse is called *chemical synapse* (or *electrochemical synapse*) (Fig. 1.3 [91]) and the space that separates these is called *synaptic gap*. The synapses considered in this work are the chemical synapses (for reasons of convenience, in the rest of the thesis we refer to these by the simple name of synapses).

In the synapses, a nerve impulse causes the release of a chemical mediator at the level of the neuritic terminal of the pre-synaptic bouton. This mediator transfers the nerve impulse to the post-synaptic neuron [66]. By means of the synapses, the signals can be transmitted from one neuron to the next: upon the activation, the synaptic bouton releases a small amount of a transmitter substance (i.e., *neurotransmitters*) that stimulates the neuron's membrane on which it rests. In this way the information comes from the pre-synaptic to the post-synaptic cell.

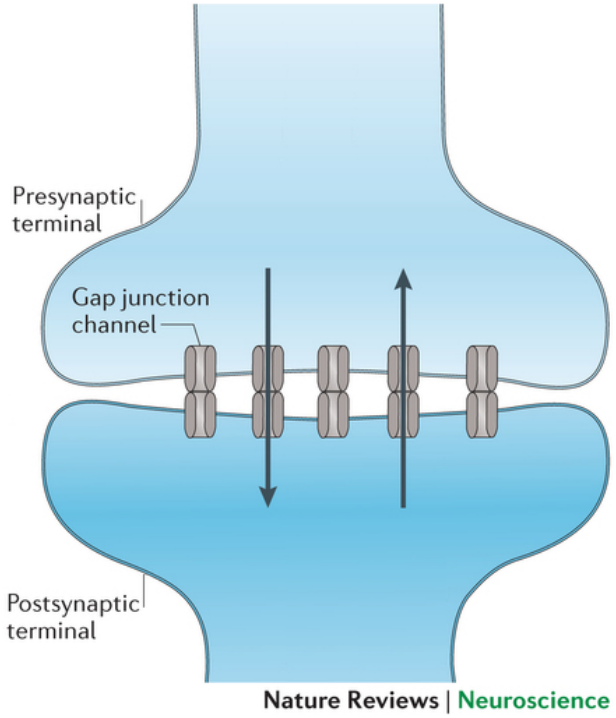

FIGURE 1.2: Representation of an electrical synapse [91].

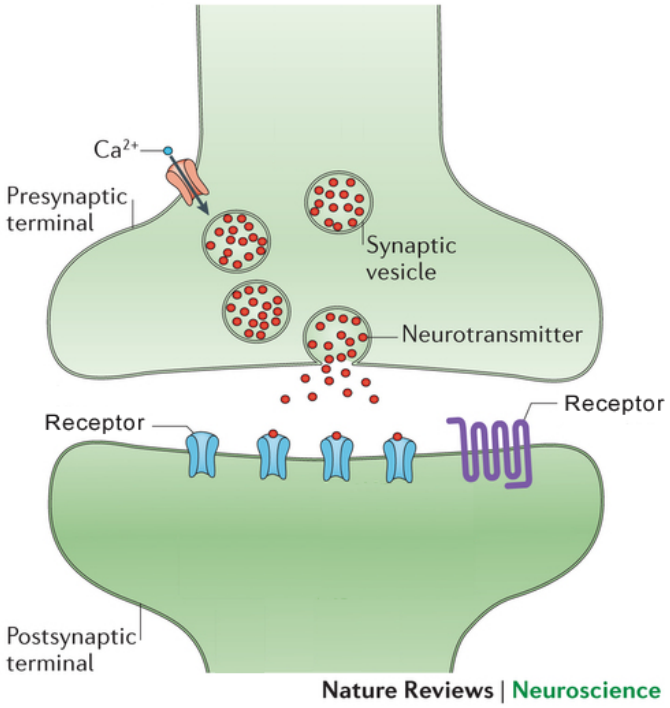

FIGURE 1.3: Representation of a chemical synapse [91].

Many axons, especially those of large dimensions, are covered by a *myelin sheath* (see Figure 1.1), which has both a protective purpose and the aim to increase the conduction velocity. This system ensures the propagation of the electrical impulses (i.e., *spikes*) along the axon. These are generated by means of a mechanism of *polarization* and

depolarization of the neuron membrane, which acts with a rocking motion.

1.2 Classification of biological neurons

It is possible to distinguish various types of neurons according to their morphology, the function that they cover and their location within the nervous system. From a morphological point of view, depending on how many extensions radiate from the cell body, neurons can be classified into 3 types: *unipolar neurons*, which are provided with only the axon; *bipolar neurons*, having only one dendrite and one axon that detach from the opposite poles of the soma; *multi-polar neurons*, so called because they have a single axon and many dendrites emerging from various points of the soma.

Moreover, according to the function and the direction of propagation of the nerve impulses, it is possible to divide the neurons into other 3 types: *afferent neurons*, which receive stimuli from the outside and send these to other neurons, by means of electrical impulses; *central neurons*, which receive electrical impulses from other neurons and, after a partial processing, they retransmit them to other neurons interconnected; *effector neurons*, which receive signals from other neurons and transmit them to the muscles, causing the contraction of these latter.

Finally, it is possible to distinguish neurons according to their positioning within the nervous system: in this regard, at the level of the cerebral cortex, the type of cell most widespread is the *pyramidal neuron* (Fig. 1.4). This has a large soma, with a triangular shape, and possesses long dendrites equipped with plugs, which are small extensions that represent post-synaptic sites. The pyramidal neuron, which from a functional point of view belongs to the category of the central neurons, is a multi-polar excitatory cell, characteristic of the *hippocampus* (a particular brain region), whose axon ends in other regions of the brain and in the spinal cord.

The main feature of these neurons is the cortical disposition: the apical dendrites cross different cortical layers and are always oriented perpendicular to the surface of the cortex. This dendritic organization facilitates the integration of the various afferent signals.

1.3 Dynamics of biological neurons

Here we show how the transmission and the propagation of the nervous information occur from a neuron to another one.

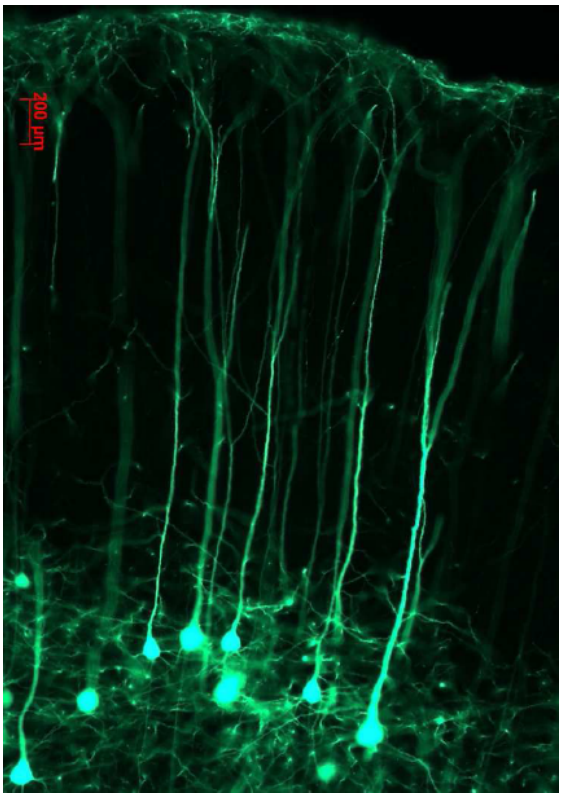

FIGURE 1.4: Pyramidal neurons of the cortical layer.

Neurons are cells with electric activity, hence there is a potential difference between the inside and the outside of the cell membrane. In particular, this latter has an electric potential V (or V_m) called *membrane potential* and measured in Volts (V). As a first approximation, we assume that V is equal at all the points of the membrane (see Figure 1.5 [103]). Moreover, assuming that the potential outside of the membrane is equal to 0, the potential when the cell is not activated (i.e., *resting potential*) is approximatively equal to $-70mV$.

The presence of such an electric potential at the neuron membrane is due to the balancing of the charges between the internal (*intra-cellular liquid*) and external (*extra-cellular liquid*) environment of the cell. Several types of ions (with positive or negative charge) are present inside and outside the cell and the difference between the inner and the outer concentration of these ions produces the polarization of the membrane [103].

Ions such as K^+ (potassium), Na^+ (sodium) and Cl^- (chlorine) are present both inside and outside the neuron membrane, but with different distributions. In details, the extra-cellular liquid contains large amounts of ions of sodium and chlorine and small amounts of potassium ions, whereas in the intra-cellular liquid the opposite occurs.

The difference of ion concentrations between the inside and the outside of the cell is preserved by the incessant work of the sodium-potassium pomp (see Figure 1.6) that,

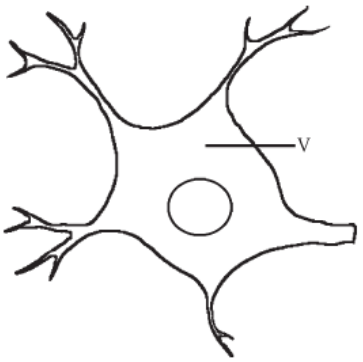

FIGURE 1.5: The membrane of the neuron is assumed, in a first approximation, to have constant electric potential V [103].

at each cycle, ejects 3 ions Na^+ and imports 2 ions K^+, consuming energy by means of the destruction of the ATP molecule (Adenosine TriPhosphate): in this way, a negative electrical potential is generated inside of the cell.

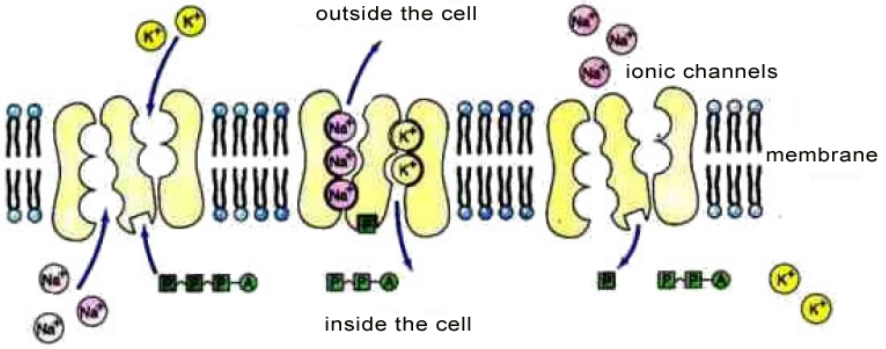

FIGURE 1.6: Sodium-potassium pump.

The cell membrane is composed of *ion channels* that, under certain conditions, selectively allow the passage of the above-mentioned ions between the inside and the outside of the membrane. There are ion channels that are sensitive to Na^+, which govern the passage of sodium through the cell membrane, and ion channels that are sensitive to K^+, that govern the passage of potassium. Moreover, the ion channels can be classified according to the speed by which the ions pass through them.

A neuron can be "stimulated" in order to induce a potential gradient that depolarizes the cell membrane, opening ion channels so that there is a flow of positive ions from outside to inside. The opening of the ion channels, in which the currents flow, is regulated by the neurotransmitters, which reach the ion channels by means of the action potentials (i.e., *spikes*) and arrive at the pre-synaptic junction. Obviously, this phenomenon is not linear. If the stimulus is greater than a certain threshold, (i.e., approximately $-55mV$), then the membrane is depolarized and the neuron emits a spike, as we can observe in Figure 1.7. More precisely, when the opening of the channels occurs, the flow of the ions

towards the inside or outside of the cell determines the depolarization (the potential increases) or the re-polarization (the potential decreases) of the neuron.

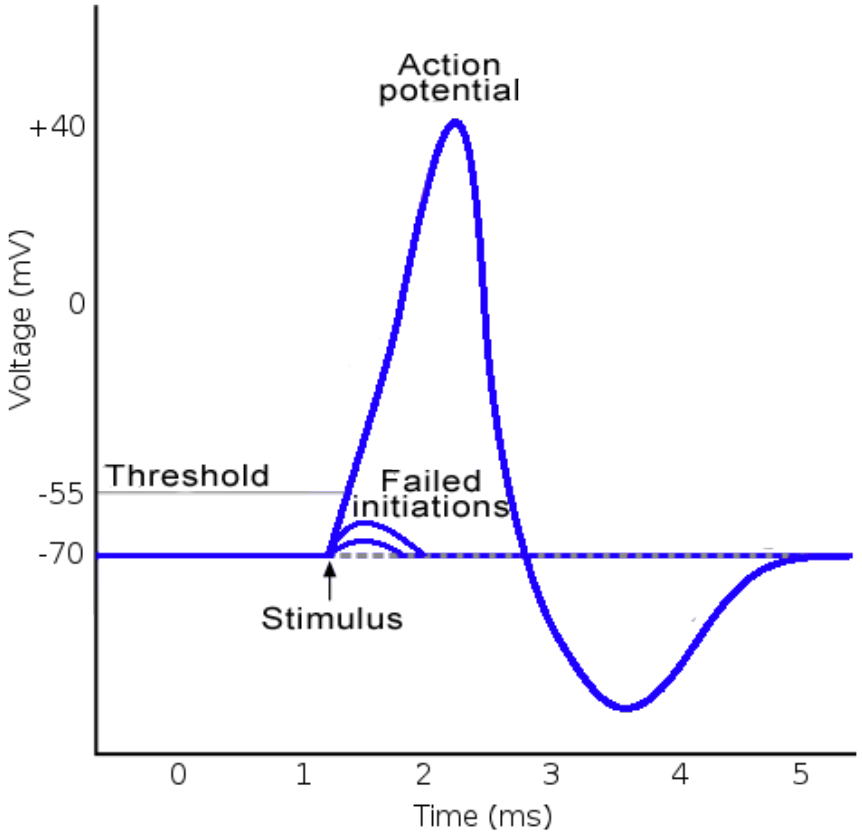

FIGURE 1.7: Schematic representation of an action potential.

Hence, the action potential consists in a rapid variation of the membrane potential: the spike starts with an abrupt change that, by the normal resting potential (i.e., approximately $-70mV$), leads to a positive membrane potential and ends with a further change, just as fast, which restores the negative potential.

As shown in Figure 1.8, in this process it is possible to distinguish several phases. The first one is the *resting state* (1 in Figure 1.8), which precedes the onset of the action potential: during this phase the membrane is polarized, because the membrane potential assumes the value of the resting potential. In the second phase, i.e. the *depolarization* (2 in Figure 1.8), the opening of the ion channels that are sensitive to the Na^+ ions occurs: consequence of this fact is the entry of a large number of Na^+ ions within the cell. The third phase, i.e., the *re-polarization* (3 in Figure 1.8), starts with the inactivation of the Na^+ channels (no other Na^+ ion enters the cell) and the opening of the K^+ channels (K^+ ions begin to leave the cell): this causes a migration of positive charges to the outside of the cell and then the membrane potential returns to resting level. At the end of this phase, Na^+ channels are closed, while K^+ channels start to close. Finally, the

fourth phase is the *hyper-polarization* (4 in Figure 1.8), which is caused by the continued movement of K^+ ions out of the cell: some K^+ channels are not yet closed, hence the potential goes even more negative than the resting value. When all the K^+ channels are closed, the membrane potential goes back to its resting value: the membrane is polarized again and the neuron returns in the *resting state*.

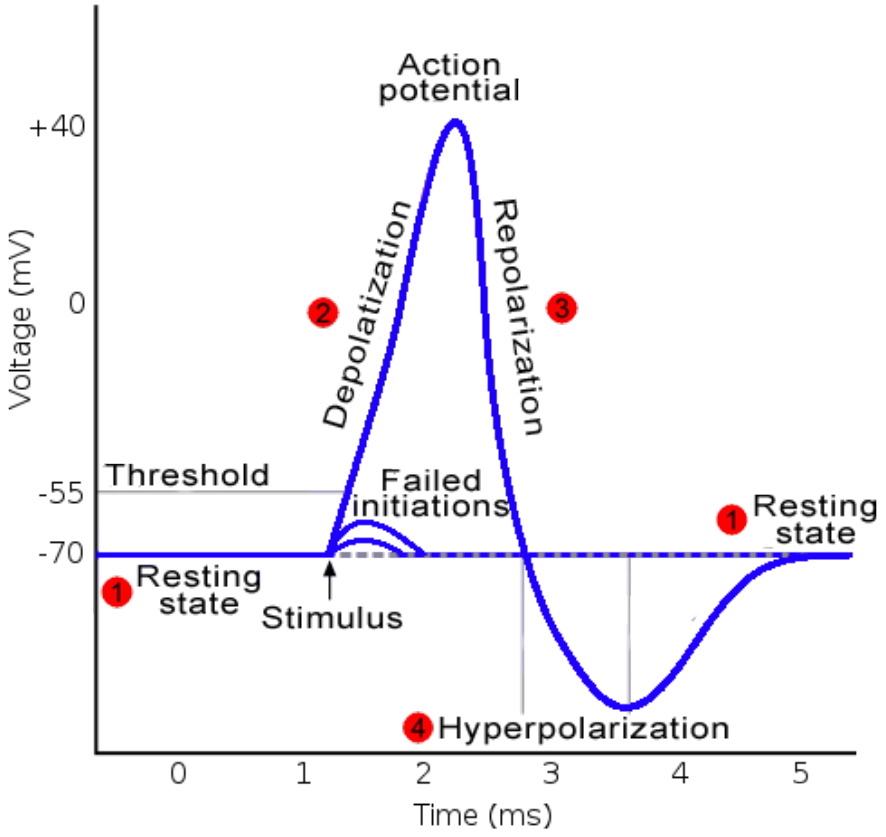

FIGURE 1.8: Schematic representation of the different phases of an action potential.

Figure 1.9 shows that in the first part of the depolarization phase there is a time period (i.e., *absolute refractory*), during which no new depolarization process can occur. Moreover, there exists also another time period (i.e., *relative refractory*), subsequent to the first one and lasting a few ms, during which a new process of depolarization can occur, but with the constraint that there is a higher threshold. In this way, a maximum "firing" frequency is determined, i.e., a frequency of consecutive action potentials, which is approximatively equal to $300Hz$.

As we can observe in Figure 1.10, a neuron is not isolated but has strong connections with the other neurons through the dendrites: in fact, the neuronal dynamics are just due to the continuous exchanges of electric currents or of charges among neurons. Hence, when a stimulated neuron emits a spike, this is propagated along the axon of the neuron

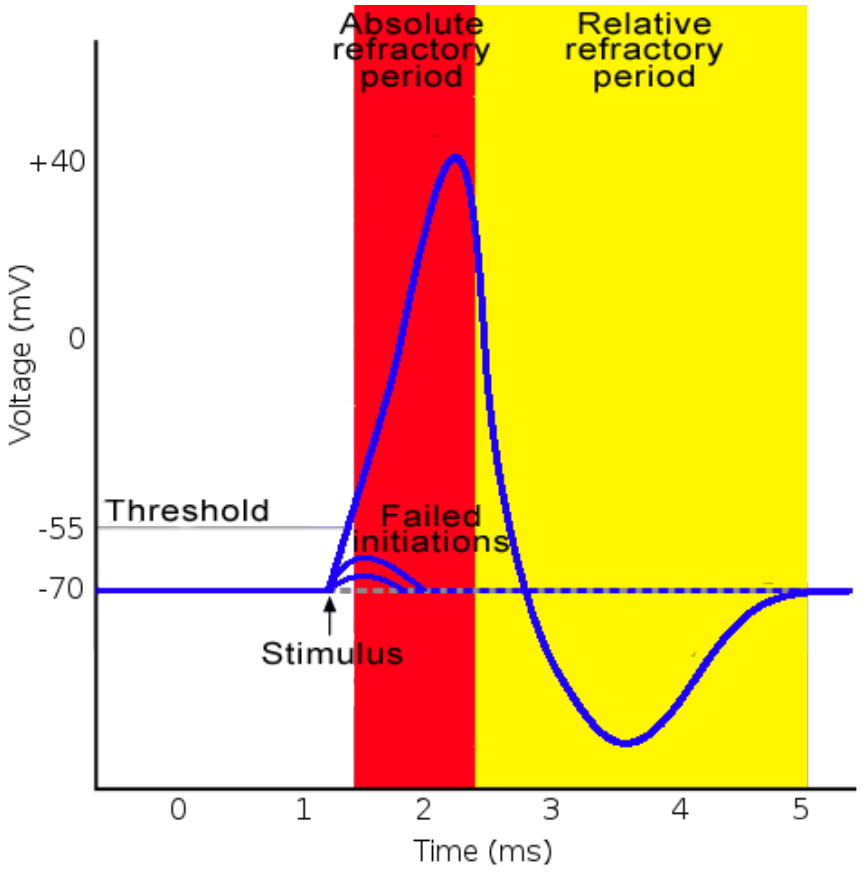

FIGURE 1.9: Schematic representation of the refractory periods.

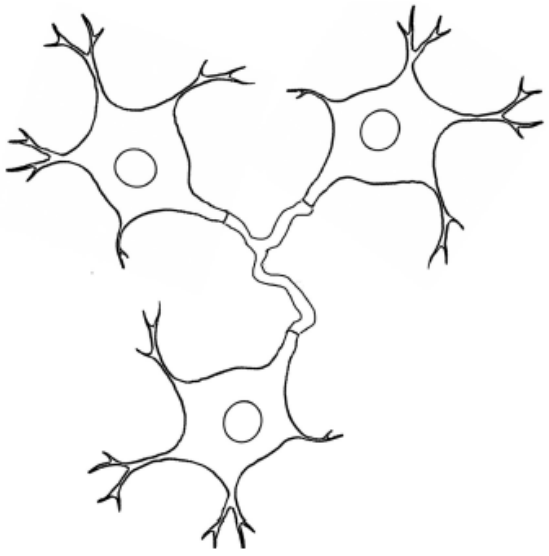

FIGURE 1.10: Schematic representation of a neural network [103].

itself, then it reaches the synaptic boutons, and, finally, it ensures that the neurotransmitters are released in the synaptic gap: in this way, the pre-synaptic neuron is able to "stimulate" the post-synaptic cells, in which there will be an action potential.

1.4 The synaptic transmission

The synaptic transmission in the central nervous system is mediated by excitatory and inhibitory amino acid neurotransmitters: the excitatory neurotransmitters (i.e., glutamate or GLU) are able to promote the creation of nervous impulses in the post-synaptic neuron; conversely, the inhibitory neurotransmitters (i.e., gamma-aminobutyric acid or GABA) are able to inhibit the impulses. Obviously, the synapses mediated by excitatory neurotransmitters are called *excitatory synapses*, while the synapses mediated by inhibitory neurotransmitters are called *inhibitory synapses* [66]. Moreover, the electrical synapses are just excitatory, while the chemical synapses can be both excitatory and inhibitory.

Glutamate activates AMPA and NMDA receptors. The AMPA receptors are channels permeable to cations, but some types are not permeable to the calcium. They have very fast kinetics of activation and inactivation. Typically, the rise time of the current varies between $0.4ms$ and $0.8ms$, whereas the time constant of the current is approximatively equal to $5ms$. The NMDA receptors are a thousand times more sensitive to glutamate than other receptors. Moreover, a peculiar feature of the NMDA receptor channels is the voltage-dependent sensitivity in order to stop the current for physiological concentrations of Mg^{2+} (magnesium). At the resting potential, each channel is obstructed by one magnesium ion coming from the outside of the cell, so that even if the receptor is activated and the channel passes in the "open" state, it does not conduct any current.

GABA activates two classes of receptors, which are GABA_A e GABA_B. The first ones have a relatively fast kinetic, whereas the second ones are slower since they involve other messengers. Moreover, the receptors are classified into other two main classes: there are ionotropic and metabotropic receptors. The first ones are themselves channels: $GABA_A$, NMDA ed AMPA are ionotropic receptors. The second ones are not channels, but they are composed by a receptor coupled to other proteins responsible for enzymatic reactions: $GABA_B$ is a metabotropic receptor.

1.5 Mathematical models representing biological neurons

To fully understand the biological dynamics characterizing the neurons, it is necessary to study the mathematical models that form the functional substrate necessary to describe them. Over the years, in order to represent the biological neurons by means of mathematical models, various attempts have been made. Although these models are not able to exactly reproduce a real system, generally it is possible to define models that are

able to exhibit a high degree of similarity with the qualitative behaviour of the system under consideration.

A mathematical model, corresponding to a particular physical system S, consists of one or more equations, whose individual solutions, in response to a given input, represent a good approximation of the variables that are measured in S. A biological model consists of a mathematical description of the cell properties, more or less accurate, and it allows to describe and predict certain biological behaviours. A neuron can be modelled at different levels of complexity: if we consider the propagation effects, then we have multi-compartmental models (see Section 1.6 for more details) defined by means of Partial Differential Equations (PDEs); on the other hand, if we assume that the action potential propagation is almost instantaneous if compared to the time scale of the generation of itself, then we have single compartment models defined by means of Ordinary Differential Equations (ODEs) and algebraic equations.

1.5.1 Model Generations

The mathematical models for biological neuron representation can be classified into 3 different generations [74]. The neuronal models belonging to the *first generation* have a great limitation: they do not consider the biological characteristics of the neuronal membrane. Hence, these models are too simple to represent the biological behaviour of the real neurons. Moreover, a characteristic feature of these models is that, for each neuron, the output signals, which are determined by means of a *threshold* "activation function", are digital, thus they can assume only two values: 1 (all) or 0 (nothing).

The *second generation* of neuronal models replaces the threshold function with a *continuous function* for computing the output signals of a given neuron: common activation functions are the *sigmoid function* $\varphi(y) = 1/(1+e^{-y})$ and the linear saturated function π with $\pi(y) = y$ for $0 \leq y \leq 1$, $\pi(y) = 0$ for $y < 0$ and $\pi(y) = 1$ for $y > 0$ [74]. Hence, the neural models from the second generation are able to compute functions with analogue input and output. These models are more realistic than those belonging to the first generation, since the output of a neuron with a sigmoid function is a representation of the current firing rate of a real biological neuron. On the other hand, as explained in [74], the "firing rate interpretation" is questionable: experimental results from neurobiology have lead to the investigation of a *third generation* of neural models that employ *spiking neurons* as calculus units.

With the *third generation* of neuronal models we can observe a clear approach to the description of biological reality, using individual spikes to describe the output signals of a cell. Obviously, these mathematical models do not provide a complete description of the

extremely complex computational function of a biological neuron [74]. In the following sections we show some of these models, which are formulated in terms of ODEs and describe the evolution of the membrane potential in function of the *time*. The third generation models have an important limitation due to the independence from *space* of the membrane potential of the neuron. Making some changes to these models (using the "*cable theory*" [49, 95]), it is possible to introduce the dependence on both time and space, as well as it occurs for biological neurons.

1.5.2 McCulloch and Pitts model

The first mathematical formalization related to the behaviour of a neuron has been proposed in [78] by McCulloch and Pitts (1943), where it is shown how simple formal neurons could be combined together to calculate the 3 elementary logical operations (i.e., NOT, AND and OR) and, starting from these, how it is possible to implement any operation of the propositional calculus.

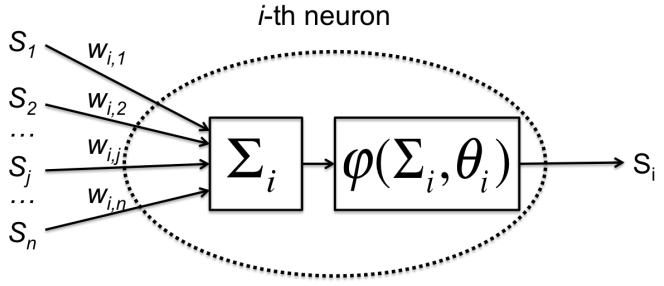

FIGURE 1.11: Schematic representation of the McCulloch and Pitts neuron.

Most specifically, as we can observe in the Figure 1.11, each i-th neuron performs the weighted sum Σ_i of the products between the states S_j (with $j = 1, \ldots, n$ and $j \neq i$) of each j-th neuron connected to it and the weights $w_{i,j}$ associated with the connections (synapses):

$$\Sigma_i = \sum_{j=1}^{n} S_j w_{i,j} \tag{1.1}$$

The value assumed by the Equation (1.1) is processed by an *activation* (or *transfer*) function φ: depending on Σ_i is greater or less than a certain predetermined neuron's activation *threshold* θ_i, the status S_i of the i-th neuron assumes the value 1 (*active status*) or 0 (*passive status*), as shown in the Equation (1.2).

$$S_i = \varphi(\Sigma_i, \theta_i) = \begin{cases} 1 & \text{if} \quad \Sigma_i \geq \theta_i \\ 0 & \text{if} \quad \Sigma_i < \theta_i \end{cases} \tag{1.2}$$

1.5.3 Integrate & Fire model

The Integrate & Fire (I&F) [67] is a simple ODE model, which is based on the idea that a neuron can be represented as a RC circuit (Fig. 1.12). Notice that, in this way, it is possible to evaluate only the effects of the membrane capacitance.

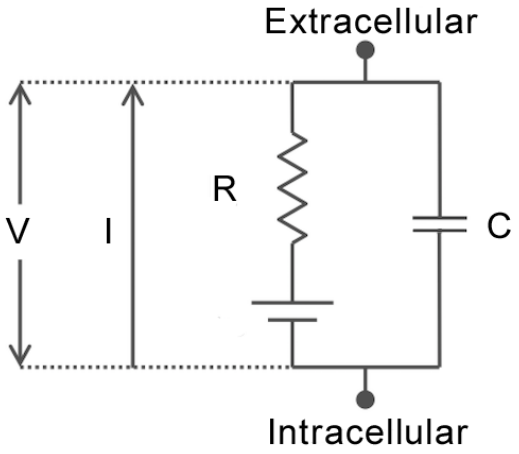

FIGURE 1.12: Elementary RC circuit as a neuron model.

The capacity C of a capacitor is the constant relating to the potential V with the charge Q of a conductor [103]

$$Q = CV \tag{1.3}$$

The Equation (1.3) is called *capacitance law* and its time derivative represents the current-voltage relationship of the capacitor, which is

$$I(t) = C\frac{dV}{dt} \tag{1.4}$$

Moreover, by the Kirchhoff's tension law, which states that the algebraic sum of the voltages acting between pairs of points in space that form any sequence closed (oriented) is equal to zero, the circuit equation is

$$RI(t) + V = 0 \tag{1.5}$$

where R is the resistance.

By replacing the Equation (1.4) in the Equation (1.5) we obtain the following differential equation with constant coefficients

$$RC\frac{dV}{dt} + V = 0 \longrightarrow \frac{dV}{dt} + \frac{V}{RC} = 0 \tag{1.6}$$

The (1.6) can be rewritten as

$$\frac{dV}{dt} + \frac{V}{\tau} = 0 \tag{1.7}$$

where $\tau = RC$ is the characteristic time of the potential.

The I&F model is not realistic in physical and biological terms. In fact, the firing frequency of the model (i.e., the inverse of the time elapsed between the occurrence of two consecutive spikes) increases linearly without bound as input current grows. The model can be made more accurate by introducing a refractory period T_{ref} (generally in the order of $1 - 2ms$), during which the neuron is insensible to external stimuli. This limits the firing frequency of the neuron by preventing it from firing during that period.

Moreover, this model implements no time-dependent memory: if the model receives a below-threshold signal at some time, it will retain that voltage boost forever until it fires again. In order to solve this problem, it is possible to add a "leak" term ($-V/R$) to the membrane potential, reflecting the diffusion of ions that occurs through the membrane when some equilibrium is not reached in the cell. Hence, the model (Eq. (1.4)) looks like

$$I(t) - \frac{V}{R} = C\frac{dV}{dt} \tag{1.8}$$

where $I(t)$ is the ionic current at time t, V is the membrane potential, R is the resistance and C is the membrane capacitance. We refer to this model as "Leak Integrate & Fire".

Finally, from the Equation (1.8), we observe that

$$I(t) = \frac{V}{R} + C\frac{dV}{dt} \tag{1.9}$$

In other words, the driving current $I(t)$ can be split into 2 components,

$$I(t) = I_R + I_C \tag{1.10}$$

where

$$I_R = \frac{V}{R} \tag{1.11}$$

and

$$I_C = C\frac{dV}{dt} \tag{1.12}$$

The first component is the resistive current I_R that passes through the linear resistor R. This can be calculated from Ohm's law ($V = IR$). The second component I_C charges the capacitor C.

Despite the Equation (1.9) makes the model more accurate than the Equation (1.4) the limit of the I&F is that there is not so much phenomenology of the neuron, since the connection of the depolarization and re-polarization of the membrane potential with the incoming or outgoing ionic currents is completely ignored [103].

1.5.4 Hodgkin-Huxley model

The Hodgkin-Huxley model (HH) [45–48] describes in detail the evolution of the membrane potential of the axon of the squid. HH is the first model with the real phenomenology of the neuron [103], which takes into account the contributions due to the ionic currents that cross the "active" channels (i.e. those which are subject to the opening/closing) and the loss effects due to the "passive" channels (called *pores*).

As well as the I&F, the HH model relies on the analogy with an electrical circuit with resistances and capacitors (Fig. 1.13). On the other hand, in this case more detailed neuronal dynamics are considered. The semi-permeable membrane of the neuron divides the interior of the cell from the outside (extracellular fluid) operating as a capacitor: any release of the input current within the cell, can lead to the loss or addition of a further charge to the capacitor through the channels of the cell membrane.

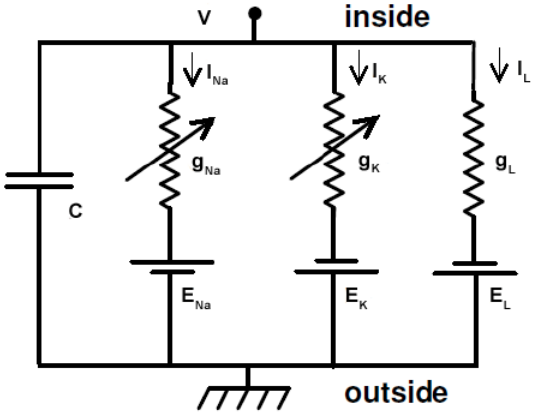

FIGURE 1.13: Equivalent electrical circuit proposed by Hodgkin and Huxley for a small segment of the squid axon.

As stated in Section 1.3, the concentration of ions inside the cell differs respect to the outside of the cell, as a result of the active transport of ions across the cellular membrane. For example, the Na^+ concentration inside the neuron is greater than that outside and the K^+ concentration inside the neuron is less than that outside. The mean motion through the membrane will be in the direction of lower concentration, thus the sodium current is going inside the neuron, while the potassium current flows outside the neuron [103]. Note that, by convention, an inward current in the neuron is considered *positive*, while an output current is *negative*.

Consequently, the current flow $I(t)$ that crosses the membrane has 2 principal components: the first one relating to the membrane capacity loading (I_{ext}) and the second one relating to the passage of specific types of ions across the membrane (I_{ion}). The ionic current is also further divided in 3 distinct components: a sodium current I_{Na}, a potassium current I_K, and a small leakage current I_L, conveyed primarily by the chloride ions (in literature, I_L is also known with I_{Cl}).

Moreover, the opening and closure of the channels is modelled through a parameter that is a function of time and satisfies a differential equation. Hence, the electrical circuit in Figure 1.13 can be described by means of the following differential equation

$$C\frac{dV}{dt} + I_{ion} = I_{ext} \tag{1.13}$$

where C is the membrane capacitance and V is the membrane potential.

We remark that, the relationship between the electric potential and the ionic current cannot follow the Ohm's law, since there are the non-linear phenomena of closure and opening of the channel. Thus, in order to describe the conductance of the channel, a non-linear function $G(V)$ is used. Then, the ionic current is

$$J_i = g_i(V - E_i) \tag{1.14}$$

Notice that the conductance is the inverse of the resistance and E_i is the reversal potential, i.e. the value of the potential such as to change the direction of the current.

So, the total ionic current is

$$I_{ion} = \sum_i J_i = \sum_i g_i(V - E_i) \tag{1.15}$$

and it represents the algebraic sum of the individual contributes relating to the types of the considered ions. Each ionic component J_i, therefore, has an associated conductance value g_i (the conductance is the reciprocal of the resistance, $g_i = 1/R_i$) and an

equilibrium potential E_i (the equilibrium potential that allows the flow of ions across the membrane is zero).

Therefore, recalling that we have assumed that the potential is the same at all points of the membrane, from the Equations (1.13) and (1.15) we obtain the equation for the current

$$C\frac{dV}{dt} = -g_{Na}(V - E_{Na}) - g_K(V - E_K) - g_L(V - E_L) + I_{ext} \tag{1.16}$$

where I_{ext} is the external current applied, the conductances g_{Na} and g_K depend on the potential and the conductance g_L is a simple constant.

The dependence on the potential is managed by means the activation parameters that are defined by

$$g_{Na} = \overline{g_{Na}} m^3 h \tag{1.17}$$

$$g_K = \overline{g_K} n^4 \tag{1.18}$$

Thus the system is defined by 4 parameters (V, m, n and h) with V and h fast variables, m and n slow variables, and with m, n and h that vary between $(0, 1)$. Moreover, n is the parameter of activation for the potassium channel, m for the sodium channel and h is the inactivation variable.

The variables m, n and h depend on the potential by means of a linear differential equation with coefficients that depend on the potential V:

$$\frac{dm}{dt} = \alpha_m(1 - m) - \beta_m m \tag{1.19}$$

$$\frac{dh}{dt} = \alpha_h(1 - h) - \beta_h h \tag{1.20}$$

$$\frac{dn}{dt} = \alpha_n(1 - n) - \beta_n n \tag{1.21}$$

with

$$\alpha_m(V) = \frac{25 - V}{10[e^{\frac{(25-V)}{10}} - 1]} \tag{1.22}$$

$$\beta_m(V) = 4e^{\frac{-V}{18}} \tag{1.23}$$

$$\alpha_h(V) = \frac{7}{100} e^{\frac{-V}{20}} \tag{1.24}$$

$$\beta_h(V) = \frac{1}{e^{\frac{(30-v)}{10}} - 1} \tag{1.25}$$

$$\alpha_n(V) = \frac{10 - V}{100[e^{\frac{(10-V)}{10}} - 1]} \tag{1.26}$$

$$\beta_n(V) = \frac{1}{8} e^{\frac{-V}{80}} \tag{1.27}$$

The values of the maximum conductance and inversion potential for each ionic channel were experimentally obtained by Hodgkin and Huxley and are reported in [48]:

$$\overline{g_{Na}} = 120 \frac{mS}{cm^2} \tag{1.28}$$

$$\overline{g_K} = 36 \frac{mS}{cm^2} \tag{1.29}$$

$$\overline{g_L} \equiv g_L = 0.3 \frac{mS}{cm^2} \tag{1.30}$$

$$E_{Na} = -115mV \tag{1.31}$$

$$E_K = 12mV \tag{1.32}$$

$$E_L = 10.599mV \tag{1.33}$$

Finally, the system of equations of the HH model is

$$\begin{cases} C\frac{dV}{dt} = -g_{Na}(V - E_{Na}) - g_K(V - E_K) - g_L(V - E_L) + I_{ext} \\ \frac{dm}{dt} = \alpha_m(1 - m) - \beta_m m \\ \frac{dh}{dt} = \alpha_h(1 - h) - \beta_h h \\ \frac{dn}{dt} = \alpha_n(1 - n) - \beta_n n \end{cases} \tag{1.34}$$

1.5.5 Modified Hodgkin-Huxley model

The HH model well describes the dynamics of the action potential for a neuron. On the other hand, it assumes that the membrane potential is the same everywhere in the cell. For real neurons this assumption is not true, because differences in potential exist along the length of this and the resulting longitudinal currents must be explicitly considered. In other words, it is necessary to abandon the hypothesis that the potential is the same at each point of the membrane.

This problem can be overcome by means of the *cable theory*, firstly applied to the conduction of potentials in an axon in [49], and then to the dendritic trees of neurons in [49, 95]. By means of the cable theory, it is possible to consider the neuron (with dendrites and axon) (see Figure 1.14 **A**) as a cylinder with uniform radius negligible with respect to its longitudinal length. In other words, the cylinder is considered unidimensional. Moreover, the cylinder is divided in portions (sub-cylinders) with equal length Δx along the x axis (see Figure 1.14 **B**). Under this hypothesis, if x is a specific point of the neuron, then $V(x, t)$ is the potential measured in the point x at time t. Finally,

the sub-cylinders, which are assumed to be iso-potential patches of the membrane, are represented by means of RC circuits that are connected in parallel (see Figure 1.14 **C**).

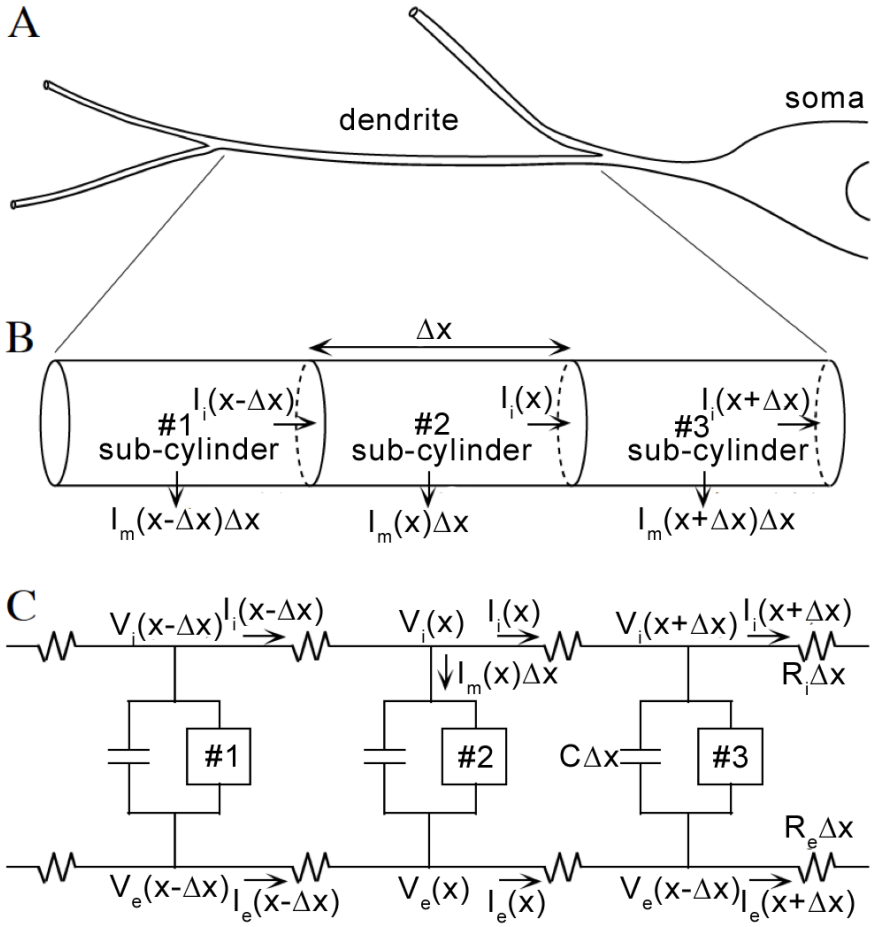

FIGURE 1.14: **A)** Sketch of a portion of the dendritic tree of a neuron emerging from the soma at right. **B)** Portion of a secondary dendrite divided into three sub-cylinders. **C)** Discrete electrical model for the three sub-cylinders.

By observing the Figure 1.14, I_m and C are the current that exits the membrane and the membrane capacitance per unit length of the cylinder, respectively. Multiplying these by Δx we obtain the total current and the capacitance in a sub-cylinder. $V_i(x)$ and $V_e(x)$ are, respectively, the membrane potentials inside and outside the cell, while $V(x) = V_i(x) - V_e(x)$ is the membrane potential. Moreover, $I_i(x)$ is the total current flowing down the interior of the cylinder and $I_e(x)$ is the total current flowing parallel to the cylinder in the extracellular space. Finally R_i and R_e are again defined as resistances per unit length of the cylinder and $R_i\Delta x$ is the resistance of the solutions inside the cylinder between the center of a sub-cylinder and the center of the next, while $R_e\Delta x$ is similarly defined as the resistance in the extracellular space between the center of two sub-cylinders.

At this point, it is possible to derive the cable equation. Considering the Ohm's law ($V = IR$) for current flow in the intracellular and extracellular spaces, we obtain:

$$V_i(x) - V_i(x + \Delta x) = I_i(x) R_i \Delta x \tag{1.35}$$

and

$$V_e(x) - V_e(x + \Delta x) = I_e(x) R_e \Delta x \tag{1.36}$$

By rearranging the Equations (1.35) and (1.36) we obtain

$$\frac{V_i(x + \Delta x) - V_i(x)}{\Delta x} = -R_i I_i(x) \tag{1.37}$$

and

$$\frac{V_e(x + \Delta x) - V_e(x)}{\Delta x} = -R_e I_e(x) \tag{1.38}$$

Moreover, taking the limit as $\Delta x \to 0$ for the Equations (1.37) and (1.38) we have

$$\lim_{\Delta x \to 0} \frac{V_i(x + \Delta x) - V_i(x)}{\Delta x} = \frac{\partial V_i}{\partial x} = -R_i I_i(x) \tag{1.39}$$

and

$$\lim_{\Delta x \to 0} \frac{V_e(x + \Delta x) - V_e(x)}{\Delta x} = \frac{\partial V_e}{\partial x} = -R_e I_e(x) \tag{1.40}$$

The conservation of current at the intracellular and extracellular nodes gives

$$I_i(x - \Delta x) - I_i(x) = I_m(x) \Delta x \quad \to \quad \frac{\partial I_i}{\partial x} = -I_m(x) \tag{1.41}$$

and

$$I_e(x - \Delta x) - I_e(x) = I_m(x) \Delta x \quad \to \quad \frac{\partial I_e}{\partial x} = -I_m(x) \tag{1.42}$$

Hence, for $V = V_i - V_e$, the membrane current I_m can be written as the sum of the current that crosses the membrane $I_R = \frac{V}{R}$ (R is membrane resistance) and the *displacement current* $I_C = C\frac{\partial V}{\partial t}$ that causes a transfer of charge from the interior of the cell to the membrane

$$I_m(x) \Delta x = \frac{V}{R} \Delta x + C \Delta x \frac{\partial V}{\partial t} \tag{1.43}$$

Moreover, differentiating and subtracting the Equations (1.37) and (1.38) with (1.39)

and (1.40), and substituting the Equations (1.41) and (1.42) allows the following relationship between the membrane potential and the membrane current to be written:

$$\begin{aligned} \frac{\partial^2 V}{\partial x^2} &= \frac{\partial^2 (V_i - V_e)}{\partial x^2} \\ &= -R_i \frac{\partial I_i}{\partial x} + R_e \frac{\partial I_e}{\partial x} \\ &= (R_i + R_e) I_m \end{aligned} \tag{1.44}$$

Finally, substituting the Equation (1.43) gives the non-linear cable equation:

$$\frac{1}{R_i + R_e} \frac{\partial^2 V}{\partial x^2} = C \frac{\partial V}{\partial t} + \frac{V}{R} \tag{1.45}$$

Now, defining $R_l = R_i + R_e$ as the longitudinal resistance and substituting in the Equation (1.45) we obtain

$$\frac{1}{R_l} \frac{\partial^2 V}{\partial x^2} = C \frac{\partial V}{\partial t} + \frac{V}{R} \tag{1.46}$$

which is a PDE of the second order.

At this point, it is possible to introduce 2 constants: the space constant λ and the time constant τ. The first one, which gives information about the distance covered by the current, is

$$\lambda = \sqrt{\frac{R}{R_l}} \tag{1.47}$$

The second one, which expresses how the potential varying the current quickly changes, is

$$\tau = RC \tag{1.48}$$

Multiplying both members of the Equation (1.46) for R and taking into account the Equations (1.47) and (1.48), and posing $V = V(x,t)$, we obtain the cable equation in its classical form:

$$\lambda^2 \frac{\partial^2 V(x,t)}{\partial x^2} = \tau \frac{\partial V(x,t)}{\partial t} + V(x,t) \tag{1.49}$$

1.6 From mathematical to computational models

The equations that describe the brain mechanisms generally do not have analytical solutions, and the intuition is not a reliable guide to understand the working of the cells and circuits of the brain [39]. For example, the Equation (1.49), which represents a modified version of the HH model, introducing the dependence on the space x, in

addition to the dependence on the time t, makes the resulting mathematical model more complicated than the HH model. In this context, the computational neuroscience community has developed several software that allow to simulate many neural models.

1.6.1 The nerve simulation environment NEURON

There exist many software that allow to simulate the neuronal dynamics. The simulations of single neuron and network models are typically based on the approach of compartmental modelling: each cell consists of many iso-potential compartments and each compartment is modelled with equations that describe electrical currents [94]. In this way, there are no restrictions on the neuronal membrane properties we can describe. This does not mean that models are "infinitely detailed", but the choice of which details to include in the model and which to omit is at the discretion of the investigator who constructs the model [39].

In Table 1.1 we report the main neuroscience software [34, 55, 63, 88, 100] and their major features. For each of these, several characteristics are analysed. More specifically, **Parallel** indicates if the software supports the parallel processing, and **Impl. lang.** represents the language with which the software is implemented. Moreover. **Progr. lang.** is the programming language used by the software, and **OS** indicates the operative systems compatible with this. Then, **GUI** indicates if it has a GUI, while **Interpreter** indicates if it has an interpreter. Furthermore, **Mod. def.** inquires by means of what is defined the model, **Ionic ch.**, if it manages ionic channels, **Syn. ch.** if it manages synaptic channels, and **Neural net.**: if it manages network models. Finally, **Int. method** indicates the integration method(s) used (*Impl*: implicit; *Expl*: explicit), **Web site** if the software has a web site, **User group** if it has a users group, and **Book**, if there are published books about the simulator.

Each of these software differs with respect to the others for the complexity of the neuronal models that it can perform: for example, Nodus is specific for the modelling of small neural networks and it is easier to use than other software, but it is slower in terms of performance. An other difference that distinguishes these software is that some of them can only be used with certain operating systems: GENESIS runs only in Unix environment, Nodus only in MAC OS environment; NEURON, instead, consists of different versions that run on Linux/Unix, MAC OS and Microsoft Windows environments.

The widely diffused nerve simulation environments are NEURON [40] and GENESIS [55]. These frameworks have large communities where the users collect and maintain databases of published computational models. In this Thesis we have chosen NEURON, which can solve problems at several levels of detail. This software is very flexible and

Question	NEURON	GENESIS	Nodus	SNNAP	Surf-Hippo	NeuronC	HHSim
Parallel	MPI	MPI, PVM	NO	NO	NO	NO	NO
Impl. lang.	C	C	Fortran	Java	Lisp	C	Matlab
Progr. lang.	Hoc, NMODL	C, Genesis scripting	NO	NO	NO	NO	NO
OS	Unix Win MacOs	Unix	MacOs	any	Unix	Unix	Win Unix Win MacOs
GUI	YES	YES	YES	YES	YES	NO	YES
Interpreter	YES	YES	NO	NO	NO	YES	NO
Mod. def.	Interpreter	Interpreter	Editor	Editor	Files	Interpreter	GUI
Ionic ch.	YES	YES	YES	YES	YES	YES	HH
Syn. ch.	YES	YES	YES	YES	YES	YES	NO
Neural net.	YES	YES	Limited	YES	YES	YES	NO
Int. method	Impl.	Expl., Impl.	Expl.	Expl.	Impl.	Exp., Impl.	No info
Web site	YES	YES	YES	YES	YES	YES	YES
User group	SenseLab	BABEL	NO	NO	NO	NO	NO
Book	YES	YES	NO	NO	NO	NO	NO

TABLE 1.1: Features of several compartmental modelling packages.

can provide many advantages: firstly, users handle directly with neuroscience concepts; secondly, NEURON provides functions that are specifically tailored to control the simulation and plot the results; thirdly, the computational engine of NEURON is very efficient because of the use of special methods for dealing with nerve equations [39]. Moreover, NEURON is an environment for creating and using empirically-based models of biological single neurons and neural networks [11]. It was developed for simulating the nerve equations with cable geometry, or complex ionic channels. This simulation environment enables the user to specify a 3D-topology, which is useful when the model is based on the anatomical reconstruction of data.

NEURON is composed by an interpreter based on the program language "`hoc`" (High Order Calculator) [59]. This language was further extended by the addition of new features, such as the Graphical User Interface (GUI). The NEURON's GUI contains several tools that can be used to construct models, exercise simulations, and analyse results. Moreover, users are able to investigate new kinds of membrane channels [39] and add new synaptic mechanisms to the neural models by using the model description language NMODL [86], which is a descendant of MOdel Description Language. These mechanisms are described as a set of non-linear algebraic and differential equations, or kinetic reaction schemes. By means of the execution of a shell script, named `nrnivmodl` under UNIX and Linux OS, and `mknrndll` under MS Windows OS, it is possible to compile and link the synaptic mechanisms of the model into a dynamically loadable library. This recalls an executable (`nocmodl`) that, starting from the NMODL files, generates the equivalent C files (`.c`), and the relative object files (`.o`), by means of the `gcc` compiler. Users can choice between two different kinds of integration methods: the first one is a first-order fully implicit integration method (backward Euler) and the second one is a more accurate second-order variant of the Crank-Nicholson time step [88].

NEURON does not deal directly with compartments, but it is designed around the notion of continuous cable "sections" that are connected together to form any kind of branched tree structure [88]. So, the basic computational task is to numerically solve the cable equation describing the relationship between current and voltage in a one-dimensional cable. The spatial discretization of this partial differential equation is equivalent to reducing the spatially distributed neuron to a set of connected compartments [39]. The notion of section makes possible to separate the biological properties from the numerical issue of compartment size [39]. In this context, each section is divided into a number of segments of equal length: this number, which can have a different value for each section [39], is denoted by the section parameter `nseg` [88]. At the center of each segment there is a *node*, which is the location where the internal voltage of the segment is defined: it is crucial to realize that the location of the second order correct voltage is not at the edge of a segment but rather at its center [39].

In NEURON, time and space (continuous variables for biological neurons) are represented by using the discretization technique, which allows to approximate partial differential equations by a set of algebraic differential equations, which can be numerically solved. The values of the continuous variable in the space and in the time are calculated on a discrete series of point in the space (i.e, the nodes) for a finite number of time instants, approximating, therefore, the continuous system to a piecewise linear system. Accordingly, in the cable Equation (1.49), the term $\frac{\partial^2 V(x,t)}{\partial x^2}$ (shown, for simplicity, with $\frac{\partial^2 V}{\partial x^2}$) is replaced by

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(x+\Delta x) - 2V(x) + V(x-\Delta x)}{\Delta x^2} \tag{1.50}$$

and the temporal derivative is replaced by

$$\frac{\partial V}{\partial t} \approx \frac{V(t+\Delta t) - V(t)}{\Delta t} \tag{1.51}$$

1.7 Pyramidal neurons of the Hippocampus

The brain is divided into several regions, each of which is responsible for different functionalities. In particular, one of the most widely studied regions of the brain is the *hippocampus*: in [1, 28, 109] it is hypothesized that the hippocampus is involved in the intermediate term storage of the memories that can be consciously recalled, i.e. the *declarative memories* [38]. The hippocampus contains two main interlocking parts: the *Ammon's horn* (or *Cornu Ammoni* (CA)) and the *Dentate Gyrus* DG. In particular, CA3 and CA1 regions of the CA part have been proposed in [104] to be, respectively, *auto-* and *heteroassociative* memories [38].

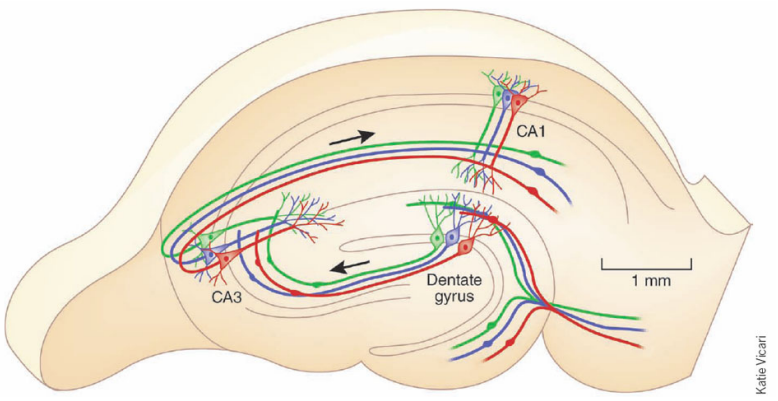

FIGURE 1.15: The hippocampus is the ragion where memory resides. Main areas are CA1, CA3 and DG.

The *associative memory* is one of the oldest Artificial Neural Network (ANN) paradigms [38]. The information managed by this are called *activity patterns*, which are representations of processed elements in the network and are stored in the memory by means of the Hebbian modification of the connections among the computing units. More precisely, in the ANNs the knowledge is stored in the strengths of connections among the neurons. Moreover, a memory is recalled when an activity pattern, which is a partial or noisy version of a stored pattern, is instantiated in the network [38] and memory recalling is characterized by how the synaptic connection strengths are plastic. A key feature of hippocampus is the *theta rhythm* $(4 - 8 \sim Hz)$, which contributes to the memory formation by separating storage and recall phases into different functional half-cycles.

The hippocampus contains two main types of cells: principal excitatory neurons, which are the main information processors of this region, and a large variety of inhibitory inter-neurons, which form connections locally [29, 102]. The excitatory neurons are the *pyramidal* cells in CA3 and CA1, and the granule cells in the DG [38]. Pyramidal neurons are so called because of the form of the soma (Fig. 1.16) and are characterized by having the axon projected to other cortical or spinal regions and spiny dendrites that cross the various layers of the cerebral cortex. Moreover, the dendrites are oriented parallel to each other and in a perpendicular way with respect to the cortex surface (cortical disposition), promoting an additions to the afferent signals. Pyramidal neurons are present in CA with different functional characteristics: CA3 neurons are activated only by certain sensory inputs, thus they have a precise spatial location and transmit signals in one directions [79]; CA1 neurons behave like CA3 neurons, but in a less accentuated way [31].

In Figure 1.15 it is possible to see that neurons belonging to the several areas of the hippocampus are interconnected, forming neural networks very complex and articulated. Moreover, DG neuron axons have synapses with apical dendrites of the CA3 pyramidal

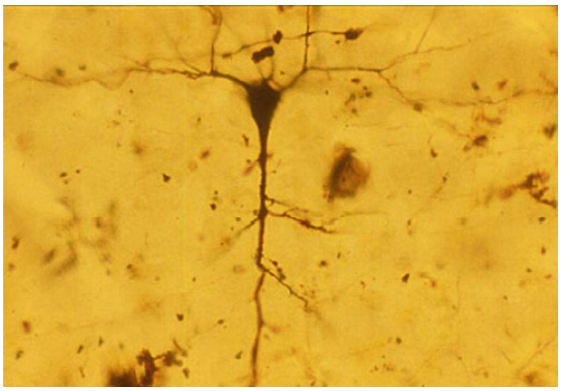

FIGURE 1.16: Pyramidal neuron of the cortical region CA1.

neurons. Finally, CA3 neuron axons have synapses with dendrites of the CA1 pyramidal neurons.

The conduction of experiments on the neural networks of the hippocampus allows us to understand the behaviour of the processes of encoding (*storage*) and retrieval (*recall*) of memories.

Chapter 2

On the mechanisms underlying the depolarization block in the spiking dynamics of CA1 pyramidal neurons

Abstract. *The prototyping and the development of computational codes for biological models, in terms of reliability, efficient and portable building blocks allow to simulate real cerebral behaviours and to validate theories and experiments. A critical issue is the tuning of a model by means of several numerical simulations with the aim to reproduce real scenarios. This requires a huge amount of computational resources to assess the impact of parameters that influence the neuronal response. In this Chapter, we describe how parallel tools are adopted to simulate the so-called depolarization block of a CA1 pyramidal cell of hippocampus. Here, high performance computing techniques are adopted in order to achieve a more efficient model simulation. Finally, we analyse the performance of this neural model, investigating the scalability and benefits on multi-core and on parallel and distributed architectures.*

2.1 Main references

[6] Bianchi, D., Marasco, A., Limongiello, A., Marchetti, C., Marie, H., Tirozzi, B., Migliore, M. *On the mechanisms underlying the depolarization block in the spiking dynamics of CA1 pyramidal neurons.*

Abstract. *Under sustained input current of increasing strength neurons eventually stop firing, entering a depolarization block. This is a robust effect that is not usually explored in experiments or explicitly implemented or tested in models. However, the range of current strength needed for a depolarization block could be easily reached with a random background activity of only a few hundred excitatory synapses. Depolarization block may thus be an important property of neurons that should be better characterized in experiments and explicitly taken into account in models at all implementation scales. Here we analyse the spiking dynamics of CA1 pyramidal neuron models using the same set of ionic currents on both an accurate morphological reconstruction and on its reduction to a single-compartment. The results show the specific ion channel properties and kinetics that are needed to reproduce the experimental findings, and how their interplay can drastically modulate the neuronal dynamics and the input current range leading to a depolarization block. We suggest that this can be one of the rate-limiting mechanisms protecting a CA1 neuron from excessive spiking activity.*

[14] **De Michele P.** et al.: Parallel tools and techniques for biological cells modelling.

Abstract. *In this paper we show a way to use high performance computing techniques in order to achieve a more suitable neural network simulation approach. Several mathematical models and software for biological cells modelling are described. The performance of a modified and optimized biological neuron computational model, based on multi-thread and message passing tools, are reported.*

2.2 Introduction

In the computational neuroscience context, many research papers (as [6, 8, 25, 92]) use parallel and scientific computing tools in order to perform simulations of complex neural models. The building up of a neural model, reproducing a real scenario, requires long and deep steps to tune several biological parameters through numerical simulations. These tasks are needed to validate the model and they represent the critical issue in terms of computational resources, both in the time and in the memory allocation. In this Chapter, we focus on the model described in [6], where a single experiment reproduction has an average execution time of $T \approx 11h$, on a "Intel Xeon E5410" CPU at $2.33GHz$ with a single core run. The tuning phase of this model requires several thousands of simulations, in order to determine the correct setting of overall biological parameters that characterize the neuron. As a result, that it is very important to find alternative strategies for reducing the impact of the simulation time in a single experiment. Generally, neuroscience papers highlight the biological results, without to deal with the computational aspects related to the parameter settings and/or to the

simulation strategies adopted; they do not report detailed analysis of the (eventually) parallel and distributed implementations of the model. Here, we analyse these aspects for the model presented in [6], based on [92, 99], where the biological phenomenon of the *depolarization block* for the CA1 pyramidal neuron of the hippocampus region is investigated. Moreover, we have formalized the computational framework underlying the model, and we present parallel tools exploited for efficient simulations. Finally, we report the performance of the proposed parallel implementation, analysing its scalability and benefits on multi-core and on parallel and distributed architectures.

2.3 Computational neural models with NEURON

To build up a computational model with NEURON, it is necessary to write numerous codes that implement several biological aspects. In Figure 2.1, we briefly report the scheme of a generic model in NEURON, which consists of four main packages.

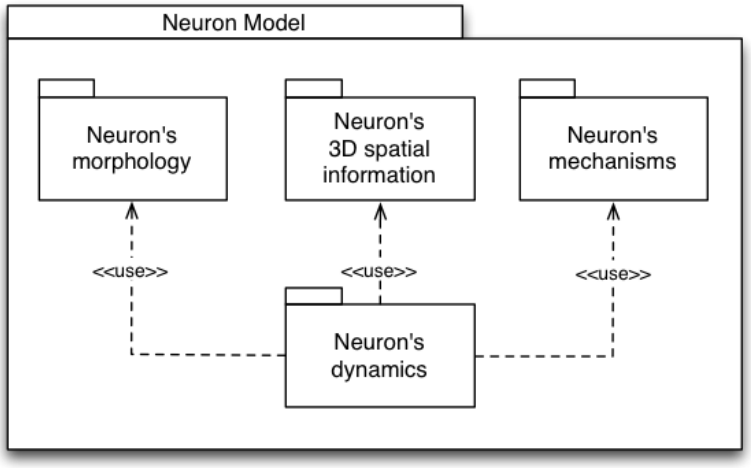

FIGURE 2.1: Model of cell in NEURON.

The first set of codes concerns the *morphology* of the neuron. In fact, as mentioned in Section 1.6.1, NEURON relies on the approach of compartmental modelling, hence the geometry of a neuron is described in terms of cylindrical sections. Each section has basic properties, which are automatically inserted by the software environment, and other mechanisms (i.e., channels) with their own properties, which can be explicitly inserted into a section. The second group of codes is needed in order to introduce 3D spatial information into the model. These codes are very useful to reuse, also in other models, the definition of the neuron in a very simple way. In order to achieve this objective, NEURON provides the *templates*. A template defines a prototype of an object from which we can also create multiple copies. The third set of codes concerns the definition of new mechanisms reproducing the dynamics of the channels characterizing a specific

neuron, by using the model description language (NMODL) (see Section 1.6.1). Finally, the fourth class of codes is adopted to define the scheme reproducing the neuron's *dynamic*, in order to replicate behaviours characterizing the real neurons. The class diagram illustrated in Figure 2.2 summarizes the representation of the first three packages of a cell model in NEURON. In Section 2.4, we will discuss about the computational framework of the model in [6].

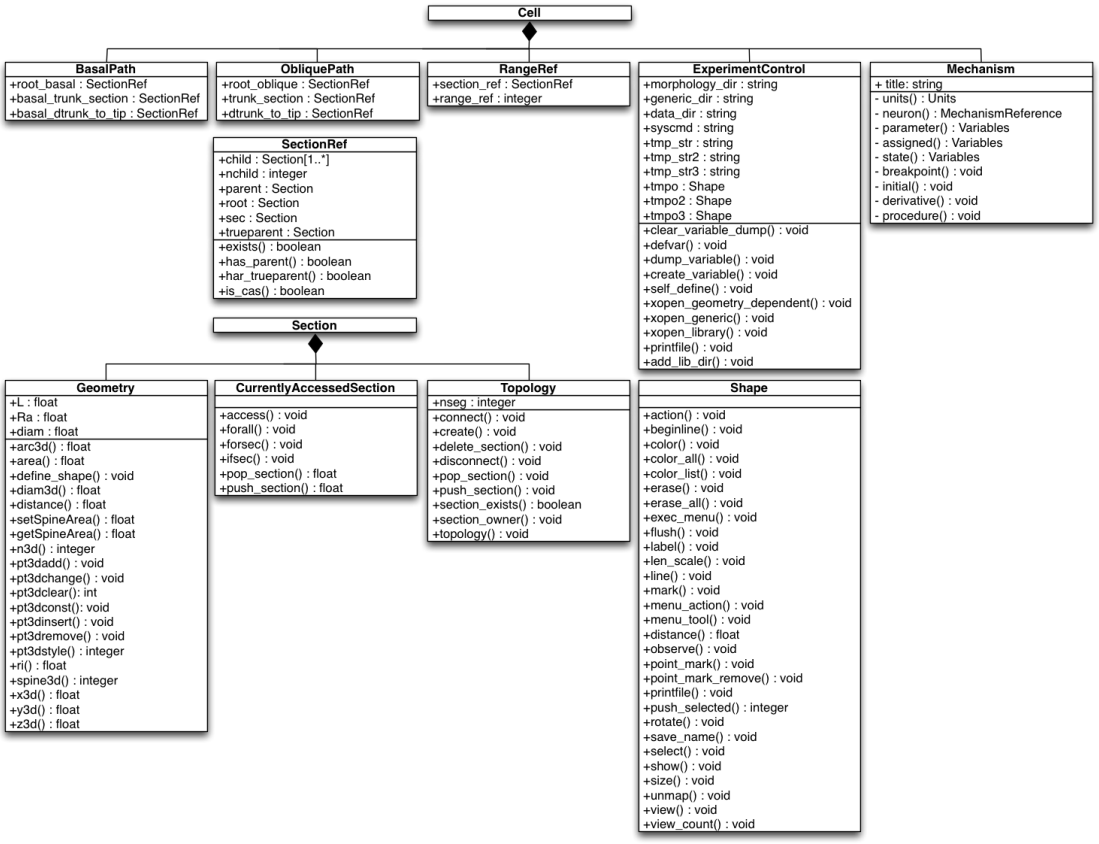

FIGURE 2.2: Class diagram of a cell model in NEURON.

2.4 A morphological model of a CA1 pyramidal neuron

In [6] a morphological model of a CA1 pyramidal neuron is presented. This model uses a CA1 pyramidal neuron implemented by merging the model in [92] (ModelDB accession number 20212), including a large set of currents experimentally observed in the CA1 neurons, and the model in [99] (ModelDB accession number 112546), including updated kinetics and distributions of dendritic channels (reviewed in [82, 83]). The authors focus on a particular behaviour of pyramidal CA1 neurons, which regularly fire, with some frequency adaptation, in response to constant increment steps of the input current I_{inj}.

Although, from the experimental point of view, the behaviour of these neurons, in relation to values of the input current I_{inj} much beyond the range of the linear increases, has not been sufficiently tested, it is well known that these cells could stop firing, reaching a *depolarization block* in which the membrane potential, during the current injection, remains constant, generally around to $-40mV$. Since very little is known regarding the way in which the models are able to reproduce the depolarization block, in [6] the authors explored the properties of the model of a CA1 pyramidal neuron, highlighting the conditions under which the neuron is able to enter in a depolarization block. The biological behaviour of the CA1 cell observed in *vivo* experiments is shown in Figure 2.3 [6]. As we can see, the synaptic activity grows its frequency with the constant increment of the input current I_{inj}, ranging from $0.3nA$ to $1.1nA$. But starting from $1nA$, it is possible to observe the depolarization block: after an initial synaptic activity, the potential stills to a constant value of $-40mV$.

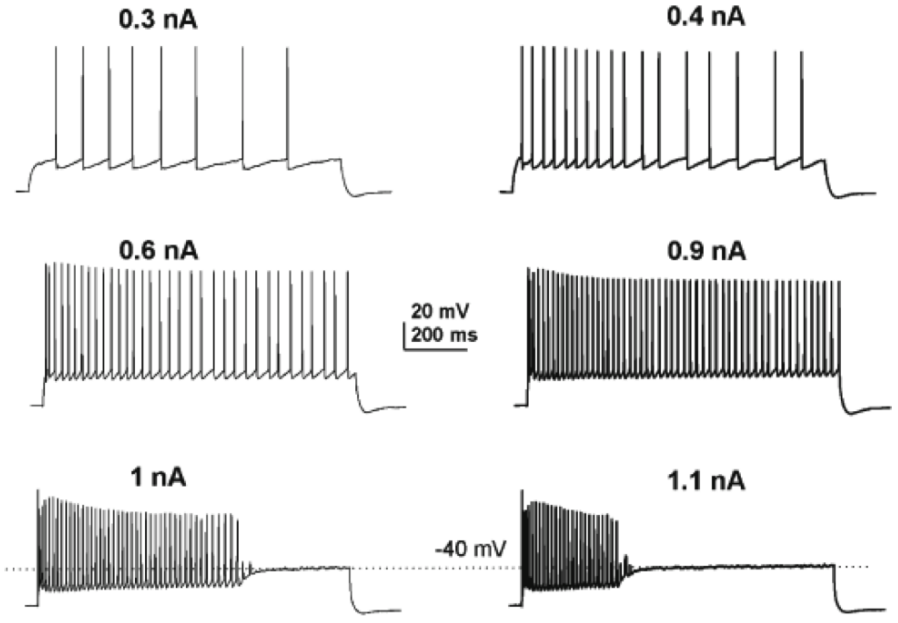

FIGURE 2.3: CA1 neurons spiking patterns in response to increasing steps for current I_{inj} from $0.3nA$ to $1.1nA$, observed in *vivo* experiments [6].

From a computational point of view, the CA1 pyramidal neuron is a complex framework requiring a lot of classes that model the neuronal morphology and the numerical dynamics. In order to reproduce a particular biological phenomenon, such as the depolarization block, it is necessary to properly set up all the parameters characterizing the CA1 model (a small part of this is shown in Figure 2.2), and the synaptic mechanisms that regulate the opening and closing of the ion channels. The first aim of the authors of [6] was that to determine which mechanism is responsible for the depolarization block. As illustrated in Figure 2.4, they proved that the Na_T and K_{DR} currents are sufficient to achieve the depolarization block.

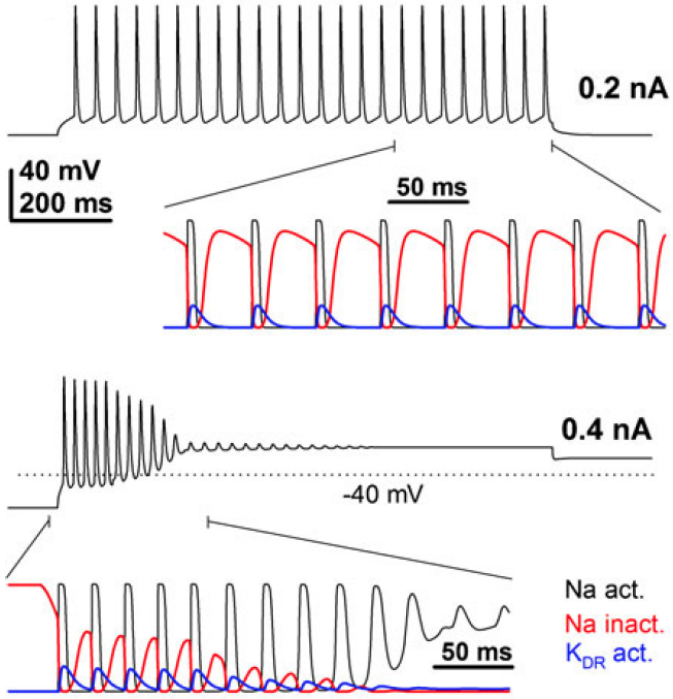

FIGURE 2.4: Model response to $0.2nA$ and $0.4nA$ depolarizing pulse of external current injected in the soma [6].

Although a suitable formulation for the Na_T and K_{DR} currents is needed to achieve depolarization block, of course these are not enough to reproduce the major experimental features that authors were interested to model in [6]. In fact, comparing Figures 2.3 and 2.4 we can observe different behaviour between the real cell and the model: this latter reproduces a depolarization block already starting from $0.4nA$, then the potential assumes constant value equal to $-20mV$. This is not compliant with the real behaviour of the cell observed in the in *vivo* experiments, in which the depolarization block occurs for current injection starting from $1nA$ and, after this, the potential assumes a constant value equal to $-40mV$. Moreover, further increasing the current injection, we can see that the depolarization block no longer occurs, as shown in Figure 2.5.

Finally, the authors of [6] have also found that both the K_M or the K_{mAHP} currents were able to reduce the potential to that observed in the in *vivo* experiments ($-40mV$), as shown in Figures 2.6 (for K_M) and 2.7 (for K_{mAHP}). Hence, when all parameters and synaptic mechanisms were set, the model has accepted all the conditions that authors have defined to characterize the experimental recordings, as shown in Figure 2.8, where it is possible to observe the same behaviour observed in Figure 2.3.

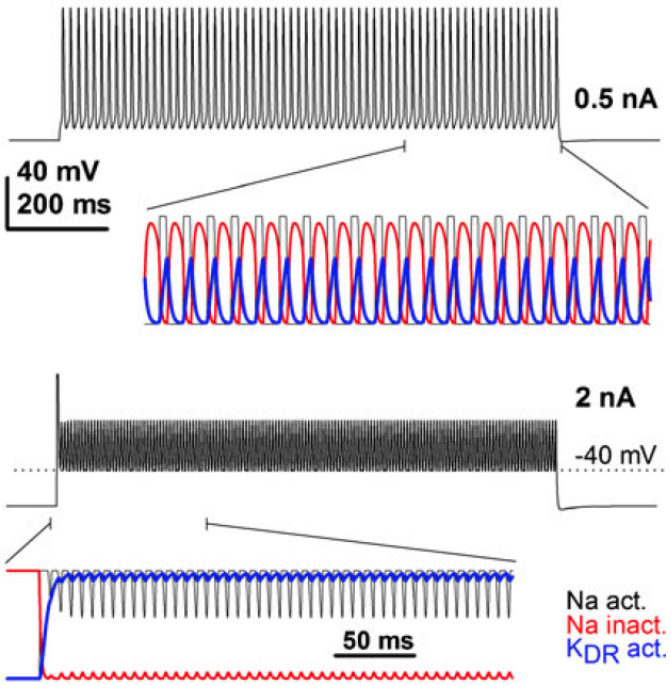

FIGURE 2.5: Model response to $0.5nA$ and $2nA$ of external current injected in the soma [6].

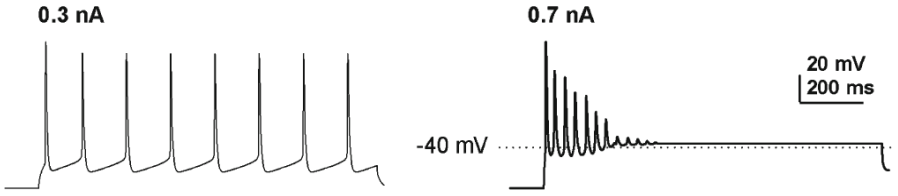

FIGURE 2.6: Model response to $0.3nA$ and $0.7nA$ of external current injected in the soma using the K_M current [6].

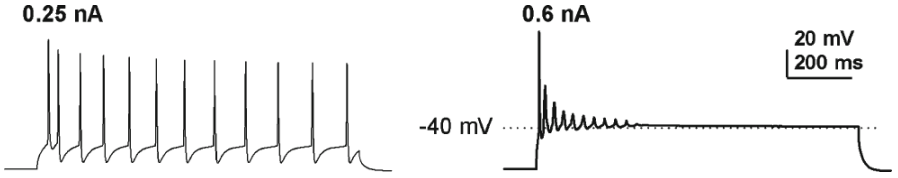

FIGURE 2.7: Model response to $0.25nA$ and $0.6nA$ of external current injected in the soma using the K_{mAHP} current [6].

2.5 Parallel tools for CA1 model

The tuning phase of the biological parameters and current mechanisms characterizing this complex model has required several thousand of simulations in order to reproduce, in the correct way, the depolarization behaviour. As just stated, a serial simulation gives execution times that are incompatible with the ability to replicate the biological phenomenon within a reasonable time-scale. In this Section we show how parallel techniques could be implemented in order to reduce the overall simulation time. As we can see in

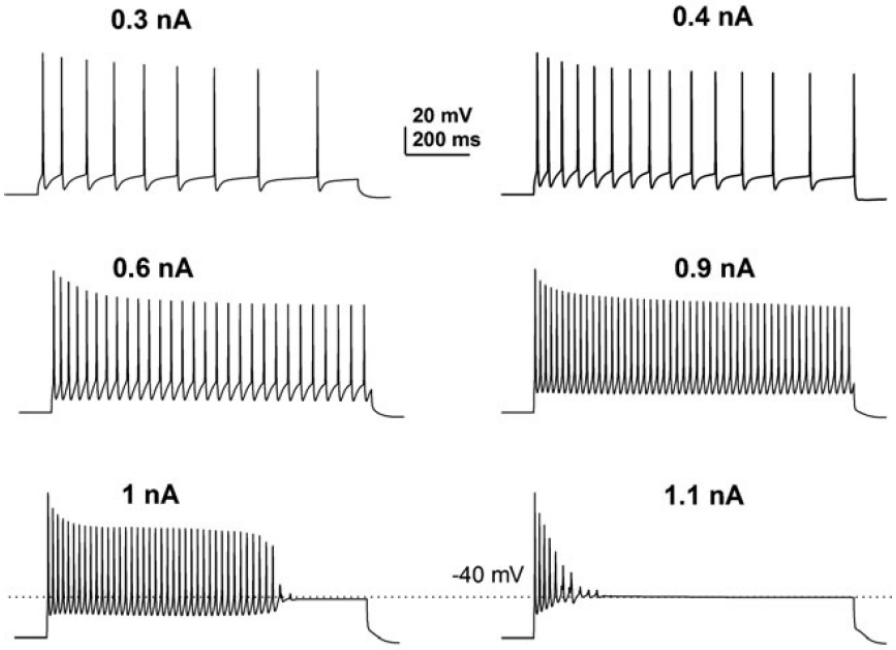

FIGURE 2.8: Model response to increasing steps for current I_{inj} from $0.3nA$ to $1.1nA$ injected in the soma [6].

Figure 2.9, we started from the *neuron's dynamic* package (in the green box), which reproduces the experiments on the CA1 cell, in order to properly set up the parameters of the *neuron's morphology* and the mechanisms of the *neuron's mechanisms* packages (in the yellow boxes). Notice that the codes in the *neuron's 3D spatial information* package remain unchanged.

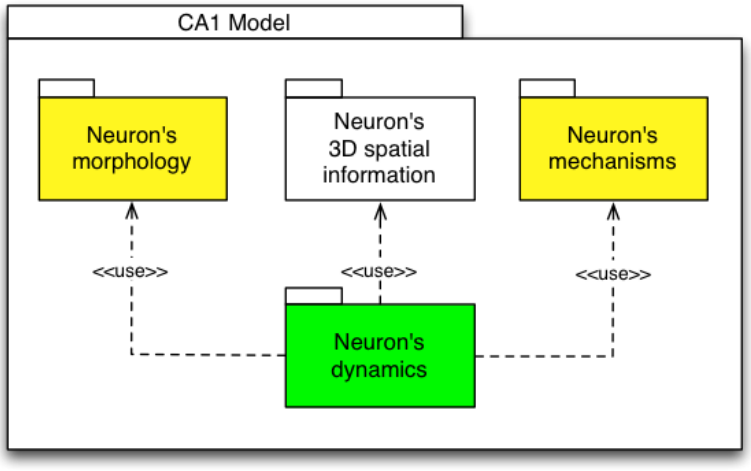

FIGURE 2.9: The packages of the CA1 model.

Figure 2.10 shows the computational tree implemented in the *neuron's dynamics* package. More in detail, firstly we created the synaptic connections stimulating the neuron with the procedures `create_syn_NMDA()` and `create_syn_GLU()`. Then, we resorted to the procedure `init()`, which is devoted to the code instrumentation. In particular, the

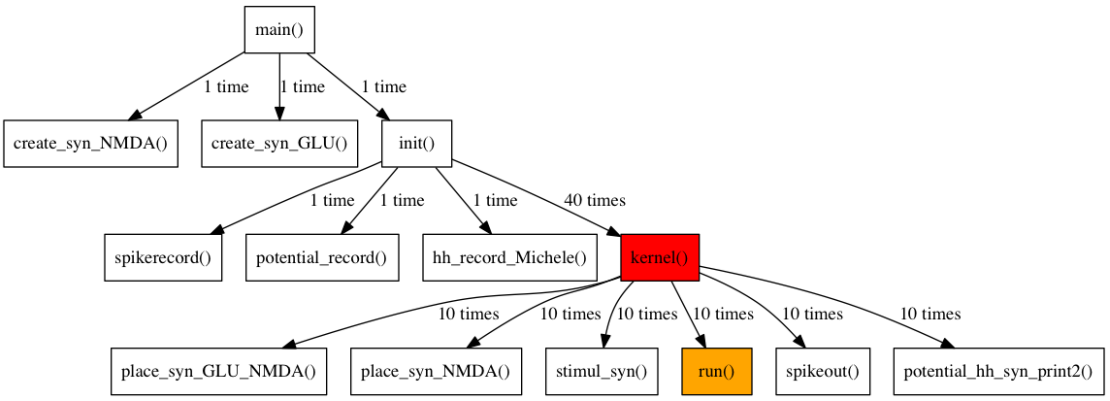

FIGURE 2.10: Procedure calls for the simulations of the CA1 model.

potential values at each time step and the times at which cell fires, are recorded by the procedures `potential_record()`, `hh_record_Michele()` and `spikerecord()`. Finally, the procedure `kernel()` (in the red box), is called (by `init()`) several times in order to reproduce the laboratory experiments. This represents the most expensive part of the package. More in detail, `kernel()` calls the sub-procedures for the placement of the synapses (`place_syn_NMDA()` and `place_syn_GLU()`), the application of the electrical stimuli to the synapses by instantiating NEURON's `NetCon` objects (`stimul_syn()`) and the firing output information (`potential_hh_syn_print2()` and `spikeout()`). `kernel()` also calls the standard procedure `run()` (in the orange box), which integrates the model, calling in turn the procedures `stdinit()` (to initialize the model invoking the native NEURON method `finitialize()`) and `continuerun()` (to perform the simulations invoking the native NEURON method `fadvance()`).

The procedure `run()`, called by `kernel()`, impacts on the $\sim$ 99% of the overall execution time, hence a crucial step is to speed up the overall simulations by changing this procedure. Here, using a parallel methodology available on the Problem Solving Environment (PSE) NEURON, we re-implemented all the *neuron's dynamics* package with new parallel codes. This PSE provides a complete suite of parallel tools, which allow for free to implement parallel strategies on a cluster of workstations or any parallel computer.

As shown in Figure 2.11, we introduced a parallel layer where the underline communication tool is the Message Passing Interface (MPI), drawn in the blue box. The yellow boxes represent the sequential part of the code, related to the sections of the cell (in the orange box) splitted among the available N processors (in the white boxes). NEURON's classes that allow the communications among the sections, resorting to MPI, are shown in the green boxes. More in details, the NEURON's *multisplit* technique [43] splits the cell into the minimum number of pieces required for load balance. This approach leads several advantages: firstly, it is not necessary to change the code defining the cell type

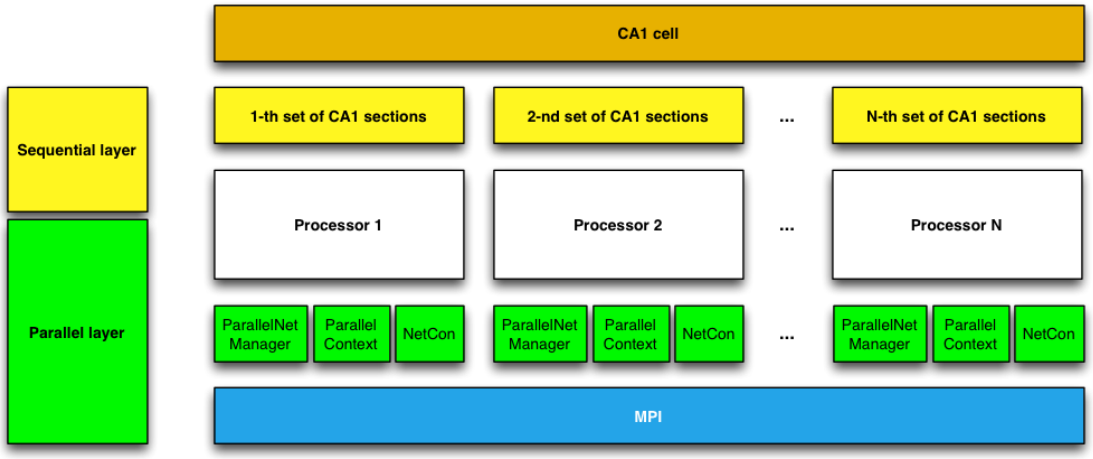

FIGURE 2.11: Parallel framework for the CA1 model.

(i.e., the implementation of the morphology, 3D spatial information and mechanisms of the neuron); moreover, it allows you to exploit, in the best possible way, the available computing resources. NEURON provides a complete suite of parallel tools, which are shown in the class diagram represented in Figure 2.12.

In the Algorithm 1 the parallel version of the procedure `kernel()`, called `P_kernel()`, is listed. We remark that the procedure `run()` is replaced by `statrun()`, which, calling the native NEURON method `psolve()`, integrates the model in a parallel context. `P_kernel()`, in order to properly invoke the procedure `statrun()`, which calls the method `psolve()` on any processor holding a set of cell sections, needs to interface with the classes `ParallelNetManager` and `ParallelContext` that manage the set up and the running of a parallel simulation. In particular, the class `ParallelNetManager` contains a collection of methods that can be used to manage the *splitted* cell. We have adapted the basic functions of the multisplit, by introducing necessary changes to its features. Obviously, the introduction of the parallelization implies the need to address several administrative problems. For example, processors can manage only the sections assigned to them. The NEURON's method `section_exists()` is used in order to check the access to various cell sections; it returns 1 if the section exists and can be used, 0 otherwise.

Remarks on the performance of the parallel application are discussed in Section 2.6.

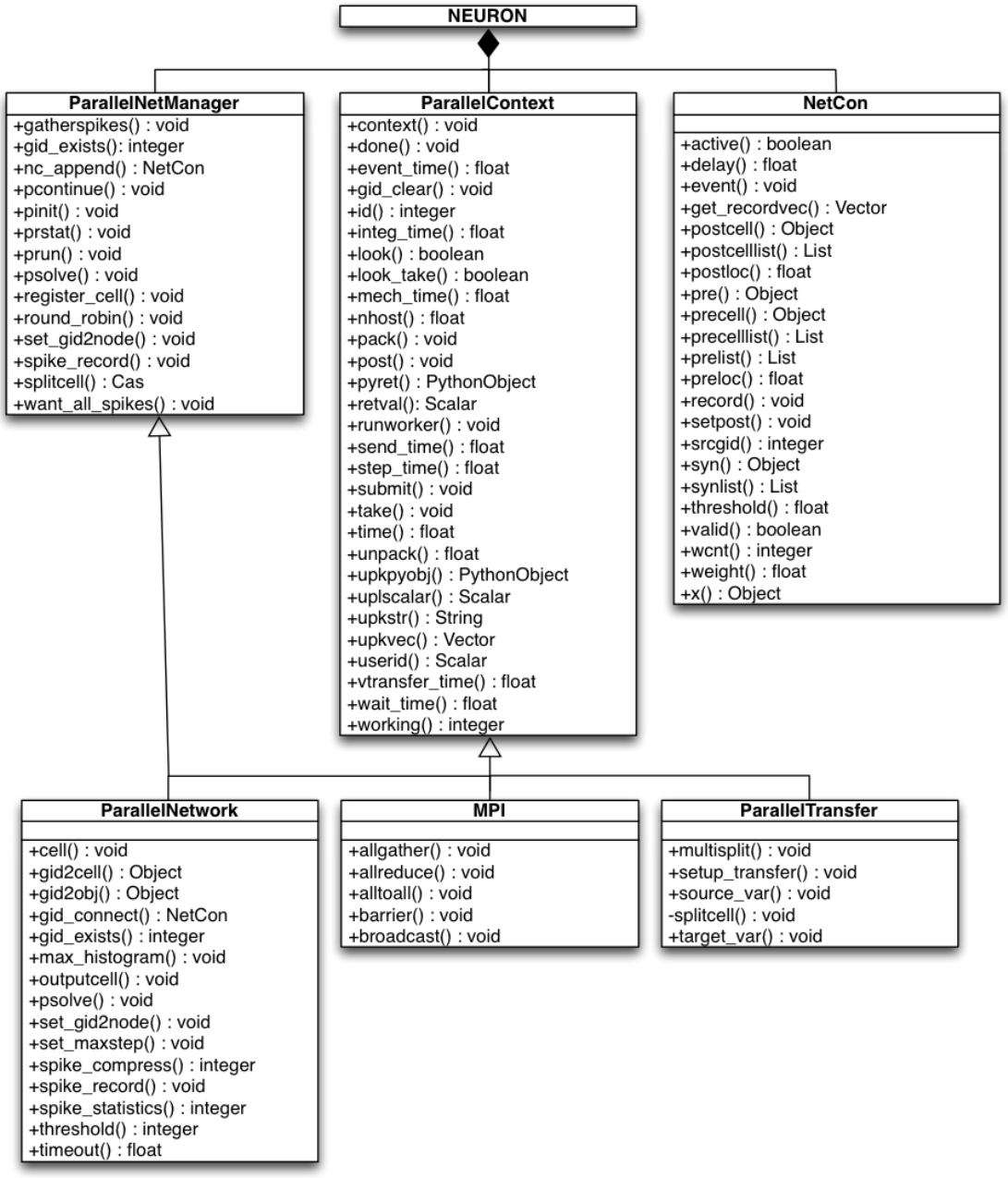

FIGURE 2.12: Class diagram representing the parallelization scheme in NEURON.

2.6 Performance results

We have performed the simulations with NEURON (version 7.1), by using the model files available on ModelDB (`http://senselab.med.yale.edu/ModelDB/`), with identification number 143719 [7]. All simulations were run with MPI using up to 32 cores on two different parallel systems: the S.Co.P.E. Grid infrastructure at University of Naples Federico II, Naples, Italy, and the CRESCO3 infrastructure of the ENEA Research Center, Portici, Naples, Italy. The S.Co.P.E. infrastructure consists of 304 blade servers each of which equipped with 2 "Intel Xeon E5410" 4−cores CPU (8−cores for each

Algorithm 1 A light view of the `P_kernel()` procedure

```
1: proc P_kernel() {
2:   ...
3:   for i_th_sim = 1, number_of_simulations {
4:    dbl_precision(32)
5:    pc.multisplit()
6:    pc.set_maxstep(100)
7:    ...
8:    /*** Synapse placement phase ***/
9:    P_place_syn_GLU_NMDA(number_of_glu_synapses)
10:   P_place_syn_NMDA(number_of_NMDA_synapses_set1,
   number_of_glu_synapses)
11:   ...
12:   /*** Synapse stimulation phase ***/
13:   P_stimul_syn(number_of_glu_synapses, number_of_nmda_synapses_set2,
   synapse_weights)
14:   ...
15:   /*** Integration phase ***/
16:   statrun(tstop)
17:   ...
18:   /*** Output information phase ***/
19:   if (section_exists(‘‘soma’’, 0))
20:    P_potential_hh_syn_print2(number_of_glu_synapses, section_name,
   i_th_sim+1)
21:   ...
22:   if (section_exists(‘‘soma’’, 0)) P_spikeout()
23:   ...
24:   pc.done()}}
```

node) at 2.33GHz (2432 cores in total), connected with 10Gbps Infiniband links, with 8 or 16GB/RAM per node. The CRESCO3 infrastructure consists of 84 blade servers each of which equipped with 2 “AMD Opteron 6234TM” 12−cores sockets (24−cores for each node) at 2.4GHz (2016 cores in total), connected with 40Gbps Infiniband links, with 64GB/RAM per node. Despite to the large number of cores available for the simulations, the choice of using up to 32 cores is related to the fact that, in general, the multisplit method well scales with test models on shared memory machines up to 8 processors and gives worthwhile reduction in runtime on 16 processors [43].

In order to evaluate how the model scales on the multi-core architectures mentioned above, in the following we report performance test results related to the number of involved cores and blade servers. In these tests, we fixed the maximum piece size factor to 0.3: thus, we split the cell in a certain number of pieces, also depending on the number of available cores, which size is at most 30% of the initial size. The multisplit technique requires several phases for the resolution of the neural model. In detail, the overall execution time (i.e., **run** time) is given by the sum of the amount of time required for

setting the split of the cell (i.e., **set up** time), the time spent by the hosts waiting for the other cores (i.e., **wait** time), the amount of time required for each core to complete its part of calculation (i.e., **step** time), and the amount of time spent by the processors to communicate with each other (i.e., the **split-cell** communication time). In Table 2.1 we show the details of the execution of the model related on several multisplit simulations up to 4 blade servers of the S.Co.P.E. infrastructure (32 cores), and in Table 2.2 we report results related to multisplit simulations up to 2 blade servers of the CRESCO3 infrastructure (32 cores). We remark that the execution times reported in Tables 2.1 and 2.2 are related to a *light execution* of the model: all the procedures described in Section 2.5 are executed just 1 time. Moreover, values for runtime (Run), set up time (Set up), wait time (Wait), step time (Step) and split-cell communication time (SCCT) are expressed in seconds (s). Finally, the speed-up (S-UP) and the relative efficiency (EFF) are reported.

From Table 2.1 we note that the model well scales on a single server (node) of the S.Co.P.E. infrastructure, moving from 1 up to 8 cores, where the speed up is 5.11 with respect to the single core test and the efficiency is equal to 64%. Moving from 1 up to 4 nodes, for a maximum number of 32 cores, we observe an high worsening of the performance. In particular, the step time remains substantially the same with 8, 16 and 32 processors, while observing slight worsening of the order of some tens of seconds, but is the split-cell communication time that affects the performance. This is due to two facts: firstly, moving from 8 to 16 and 32 cores, the memory is not shared, hence the communications have a tremendous impact on the performance; secondly, as we can see in Figure 2.13, the load imbalance becomes too high, thus, further increasing the number of cores, we can only observe a performance degradation. More in detail, the load imbalance is computed by NEURON using the following rule

$$100 \times \mathrm{cmax} \times \frac{\mathrm{nhosts}}{\mathrm{c}} - 100$$

where *cmax* is the maximum complexity (equal to ~ 6640) for a specific processor, *nhosts* is the number of processors used in the simulation and c is the total complexity of the model (equal to ~ 17500).

Servers×Cores	**Run**	**Set up**	**Wait**	**Step**	**SCCT**	**S-UP**	**EFF**
1×1	2396	0	0	2396	0%	1	1
1×8	469	0.83	0.001	451	17	5.11	0.64
2×8	969	0.97	0.004	466	502	2.47	0.15
4×8	2101	1.99	0.014	471	1628	1.13	0.04

TABLE 2.1: Performance in function of the number of involved cores for a multisplit simulation up to 4 blade servers of the S.Co.P.E. infrastructure.

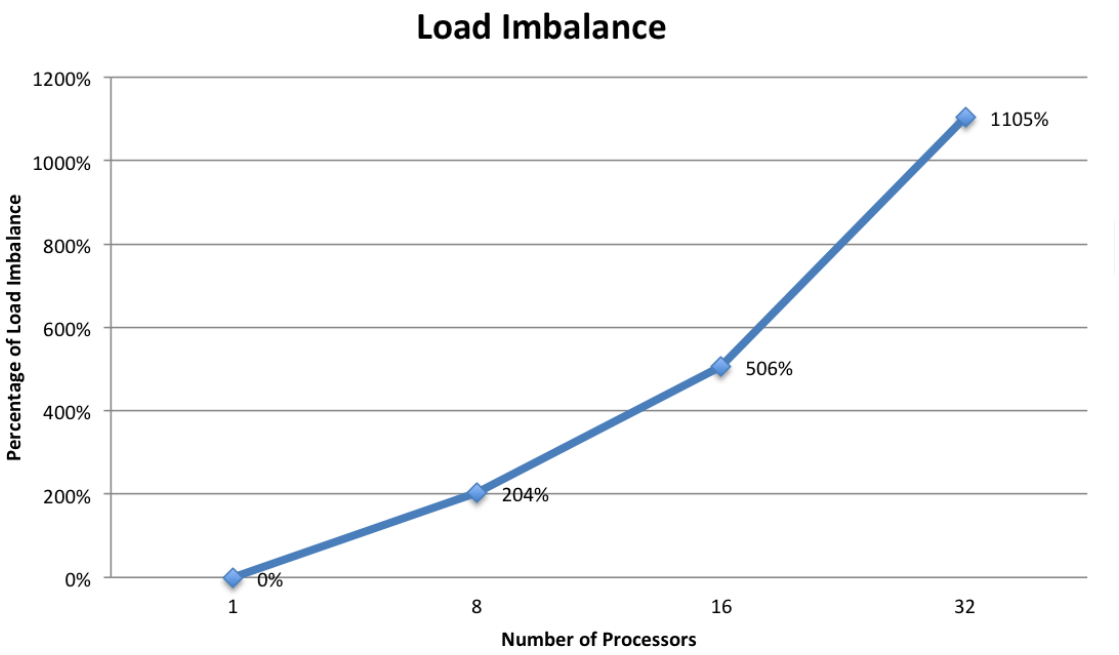

FIGURE 2.13: Percentage of load imbalance related to the complexity of the model in function of the number of processor used.

Also from Table 2.2 we observe that the model scales on a single server (node) of the CRESCO infrastructure, moving from 1 up to 16 cores. In detail, with 8 cores the speed up is 4.28 with respect to the single core test and the efficiency is equal to 54%. Moreover, with 16 cores the runtime is 4.34s and the efficiency is 15%. This fact confirms what has been said before: the shared memory among the cores positively impacts on the performance. On the other hand, increasing the number of cores gives a load imbalance too high (see Figure 2.13) and this fact negatively impacts on the performance. In particular, the step time remains the same, and the split-cell communication time decreases in the order of about ten seconds. Moving from 1 up to 2 nodes, for a maximum number of 32 cores, we observe a worsening of the performance, due to the load imbalance and to the communications among the cores.

Servers×Cores	**Run**	**Set up**	**Wait**	**Step**	**SCCT**	**S-UP**	**EFF**
1×1	1	0	0	1764	0	1	1
1×8	412	0.68	0	373	38	4.28	0.54
1×16	406	0.75	0	376	29	4.34	0.27
1×24+1×8	425	0.98	0.001	362	62	4.15	0.13

TABLE 2.2: Performance in function of the number of involved cores for a multisplit simulation up to 2 blade servers of the CRESCO3 infrastructure.

Finally, we also performed other tests in order to evaluate the impacts of a variation of the maximum piece size factor. In some cases, we observed slight improvements of the performance, but not such as to considerably impact the speed ups.

2.7 Conclusions

The simulation of biological neurons is a challenging application from a computational point of view. The calibration and the set up of a neuronal cell require mathematical models in order to simulate the biological behaviour, dependent of the cell type, and sophisticated programming environments for developing simulation codes. In the practice, building up a model that mimics the real behaviour of a biological cell, requires algorithms and communication strategies computationally expensive. In this Chapter we proposed a parallel implementation of an existing CA1 cell model, adopted to simulate the depolarization block effect. Our aim has been to formalize the computational model and to analyse the performance of the proposed parallel code on a multi-core architecture. We have observed that the main problem that has to be overcome is the communication among the compartments of a cell in a parallel simulation environment. We remarked that the cell model does not well scale on a cluster computing environment, when the spike communications are not carried out on a dedicated bus. Thus, we strongly suggest to resort to general-purpose simulation environments that support parallel multi-core programming, as massively parallel GPU architectures. In future research lines we will investigate the compatibility of NEURON with GPU computing, which it is not available at this time.

Chapter 3

Effects of increasing CREB-dependent transcription on the storage and recall processes in a hippocampal CA1 microcircuit

Abstract. *Although the building up of models of single biological neuron is very interesting to understand neuronal dynamics and behaviours, it is through the implementation of neural network models that we can study the effects of how the synaptic activity of a cell are reflected on the other neurons. Neural networks consist of several kinds of cells interconnected among them. Thus, the tuning of a large number of biological parameters and synaptic mechanisms of several different cells that belong to the network, is a more critical issue than in the case of the models of single neurons. As a result, the correct setting of the parameters characterizing a network model is tremendously expensive, from a computational point of view. In this Chapter, we present the effects of increasing cAMP Response Element Binding protein (CREB)-dependent transcription on the storage and recall processes in CA1 microcircuit of the hippocampus, and we describe how parallel tools can be adopted to simulate biological behaviours experimentally observed. Here, we resort to a parallel computing strategy in order to achieve efficient and reliable simulations. Finally, we analyse the performance of the proposed model, investigating the scalability and benefits on multi-cores and on parallel and distributed architectures.*

3.1 Main references

[8] **De Michele P.** et al.: *Effects of increasing CREB-dependent transcription on the storage and recall processes in a hippocampal CA1 microcircuit.*

Abstract. *The involvement of the hippocampus in learning processes and major brain diseases makes it an ideal candidate to investigate possible ways to devise effective therapies for memory-related pathologies like Alzheimer's Disease (AD). It has been previously reported that augmenting CREB activity increases the synaptic Long Term Potentiation (LTP) magnitude in CA1 pyramidal neurons and their intrinsic excitability in healthy rodents. It has also been suggested that hippocampal CREB signaling is likely to be down-regulated during AD, possibly degrading memory functions. Therefore, the concept of CREB-based memory enhancers, i.e. drugs that would boost memory by activation of CREB, has emerged. Here, using a model of a CA1 microcircuit, we investigate whether hippocampal CA1 pyramidal neuron properties altered by increasing CREB activity may contribute to improve memory storage and recall. With a set of patterns presented to a network, we find that the pattern recall quality under AD-like conditions is significantly better when boosting CREB function with respect to control. The results are robust and consistent upon increasing the synaptic damage expected by AD progression, supporting the idea that the use of CREB-based therapies could provide a new approach to treat AD.*

[20] **De Michele P.** et al.: *A Performance Evaluation of a Parallel Biological Network Microcircuit in NEURON.*

Abstract. *A critical issue in biological neural network modelling is the parameter tuning of a model by means of the numerical simulations to map a real scenario. This approach requires a huge amount of computational resources to assesses the impact of every model value that, generally, changes the network response. In this paper we analyse the performance of a CA1 neural network microcircuit model for pattern recognition. Moreover, we investigate its scalability and benefits on multicore and on parallel and distributed architectures.*

3.2 Secondary references

[15] **De Michele P.** et al.: *A CUBLAS-CUDA Implementation of PCG Method of an Ocean Circulation Model.*

[16] **De Michele P.** et al.: *Inverse preconditioning techniques on a GPUs architecture in global ocean models.*

[17] **De Michele P.** et al.: *An inverse preconditioner for a free surface ocean circulation model.*

[18] **De Michele P.** et al.: *A smart GPU implementation of an elliptic kernel for an ocean global circulation model.*

[19] **De Michele P.** et al.: *A regularized MRI image reconstruction based on Hessian penalty term on CPU/GPU systems.*

[90] **De Michele P.** et al.: *3D Non-Local Means denoising via multi-GPU.*

[23] **De Michele P.** et al.: *3D data denoising via nonlocal means filter by using parallel GPU strategies.*

3.3 Introduction

The step immediately following the implementation of models of single neurons is the building up of neural network models, in order to understand how the synaptic activity of a specific cell is reflected on other neurons. As stated in Chapter 2, there exist several neuroscience papers describing the biological phenomena reproduced by means of computational models, but these rarely describe the computational aspects to be taken into account for the implementation of these latter. It is clear that if the building up of a model of single neuron is very expensive from a computational point of view, then the implementation of a neural network model is even more onerous. Accordingly, in the neuroscience literature, it is easier to find models of single neuron rather than neural networks. In fact, comparing the results carried out by a search on ModelDB (`http://senselab.med.yale.edu/ModelDB/`), a reference database for computational neuroscience models, we can observe that the number of neural network models is almost half compared to those of single neurons (265 vs. 416). In this Chapter, we refer to the biological neural network model described in [8] and published in [9], for studying and understanding the effects of increasing CREB-dependant transcription on the storage and recall processes, also focusing on the computational aspects and solutions needed to properly tune the overall biological parameters characterizing this network model.

3.4 Biological overview

In the process of memory formation, storage and recall is a central, and to a large extent not understood, task carried out by the brain. A better understanding of the underlying processes is crucial to devise effective therapies for memory-related pathologies

like Alzheimer's Disease (AD). An important, but difficult, aspect in this context is to integrate the discoveries made at the cellular level during memory formation into the higher order, system-level, organization of the neuronal networks that encode memories. Although it is well known that encoding of facts and events (declarative memory) first takes place in the hippocampus [12], the molecular alterations of CA1 pyramidal neurons that result from experience- or learning-dependent synaptic activation are complex and still not fully identified.

It is now clear, however, that robust activation of CA1 synaptic inputs by CA3 axonal stimulation, leading to long-term potentiation (LTP) of the AMPA receptor current at these synapses, occurs during memory formation [108]. This robust activation of CA1 inputs results in calcium entry in the post-synaptic neuron and activates a variety of transcription factors [35], the most studied of which is the cAMP Response Element Binding protein (CREB). Recent work has characterized CREB-dependent neuronal alterations in some detail (see [3] for a review). In particular, CREB-activated neurons display a higher number of synapses containing only NMDA receptors (silent synapses), more spines, a higher magnitude of LTP, and increased excitability [4, 27, 51, 73, 77]. Furthermore, it has been recently shown that increasing CREB activity in the CA1 or dentate gyrus regions of the hippocampus enhances memory formation in rodents [96, 97].

These results suggest that CREB-dependent transcription of specific genes must, at least in part, drive memory encoding [56, 72] through a number of neuronal adaptations mediated by CREB-dependent gene transcription. By exploiting pharmacological manipulation of these adaptations, one could improve memory processes impaired by dysfunctions or diseases [10, 105]. The problem of how CREB-dependent neuronal alterations in synaptic strength, excitability, and LTP (as observed at the single neuron level using biological tools), can improve memory formation has so far escaped detailed investigation. In this paper, we have implemented a simplified CA1 network to investigate how and to what extent different cell properties, altered by increasing CREB-dependent transcription, may contribute to improve or rescue stored memory patterns under control and pathological conditions.

3.4.1 Neurons and network architecture

Our network reconstruction and its operation are based on the ideas discussed in several papers, originating from the work in [38] and recently implemented, using several important experimental constraints, in [25]. The network consists of 100 CA1 neurons with 4 types of inhibitory inter-neurons (1 iOLM, 2 iB, 1 iBS and 1 iAA), and is schematically

illustrated in Figure 3.1 **A**. Each kind of inter-neuron has a specific function in modulating not only the overall network function, but also the I/O properties of the principal neurons and, especially, the synaptic plasticity processes leading to memory storage. For the iOLM, iB, iBS and iAA we used the models implemented in [25] (ModelDB acc.n. 123815). Each inter-neuron is connected to all CA1 neurons in the appropriate dendritic/somatic/axonal region (iOLM in distal dendrites, iB in soma, iBS in proximal dendrites and iAA in axon). The network has three main input signals: excitatory inputs coming from EC (20 inputs) and CA3 Schaffer collateral (100 inputs), and inhibitory signals coming from the septum (10 inputs). The two excitatory pathways (EC and CA3) represent preprocessed sensory information and direct cortical contextual input from internal memory. They are modelled with bursts of synaptic activations at an average frequency in the gamma rhythm range (30-80 Hz), riding on top of a theta rhythm ($4 - 8Hz$), as suggested by both experimental [30] and theoretical [69] works. The EC input activation precedes CA3 inputs by $9ms$ on average, consistent with experimental data [68, 101]. In detail, EC cells were modelled as independent noisy spike trains, using a pre-synaptic spike generator. A spike train consisted of spikes at an average gamma frequency of $40Hz$, with spike times Gaussian-distributed around an average ISI of $25 \pm 0.2ms$. CA3 pyramidal cells were modelled as spike trains of the same form and with the same characteristics (mean frequency and noise level) as the EC cells. Each CA3 input makes two synaptic connections on the proximal dendrite of all CA1 neurons (one AMPA and one NMDA synapse), and each EC input forms two synaptic contacts on distal dendrites (AMPA synapses) with 20 different, randomly selected, CA1 neurons. The septal input is modelled as bursts of synaptic activations at a mean frequency of $50Hz$ and length equal to a third of a theta cycle.

CA1 neurons had the same morphology template used in [25] but with different distributions of ionic currents. Therefore each cell was implemented with 15 compartments: soma, axon, two sections for basal dendrites (oriens proximal and distal compartments), three sections for trunk (radiatum proximal, medial, distal compartments) and three sections for distal dendrites for each of the two branches (lacunosus moleculare thin, medium and thick compartments). Neurons included a transient Na^+ current, potassium K_{DR}, K_A, and K_M currents, the non-specific I_h current, three main types of calcium currents (Ca_T, Ca_N, and Ca_L), slow and fast calcium-dependent K^+ currents (K_{AHP} and K_C), and a calcium extrusion mechanism. All currents were taken from a recent model [6] (ModelDB acc.n. 143719), and reproduce the depolarization block observed in these neurons. The calcium currents were distributed according to the experimental findings [76], the K_A and I_h increased linearly with distance from the soma [50, 75], and the K_{AHP} and K_C currents were uniformly distributed. The peak conductances of the ionic currents were adjusted to qualitatively reproduce specific experimental

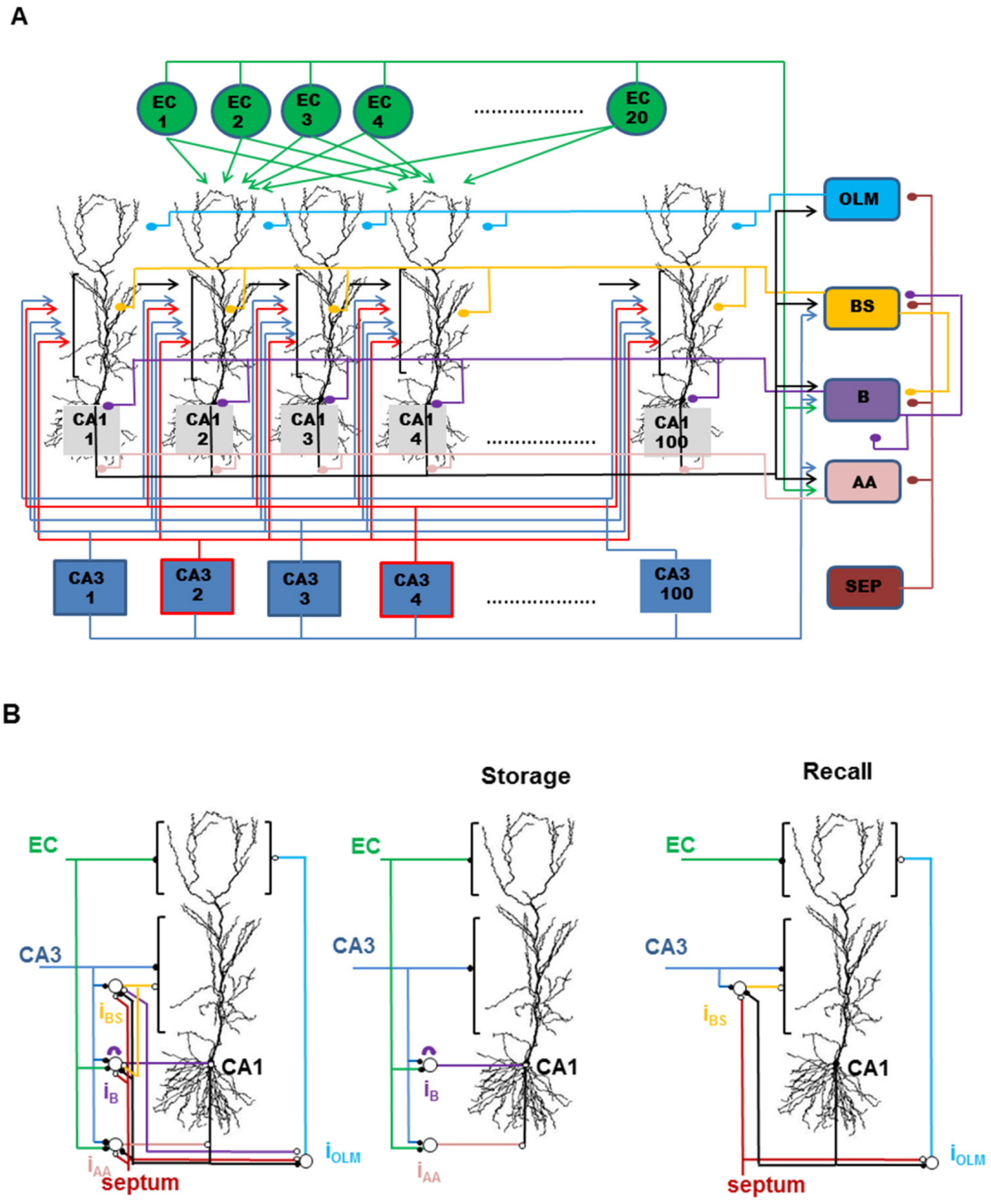

FIGURE 3.1: Diagram of the hippocampal CA1 microcircuit used in all simulations. **A)** Schematic representation of cell types and their connectivity; arrows and small ovals represent excitatory and inhibitory connections, respectively; EC: entorhinal cortex input; CA3: Schaffer collateral input; AA: axo-axonic cell; B: basket cell; BS: bistratified cell; OLM: oriens lacunosum-moleculare cell; SEP: Septal GABA input; active CA3 inputs are represented by a red outline. **B)** Schematic representation of the synaptic connections on a CA1 neuron; (left) All connections, (middle) active connections during a storage cycle, (right) active connections during a recall cycle. Black and white circles represent excitatory and inhibitory connections, respectively.

findings [73] in control and increased CREB conditions. Both the relative cell proportions and the connectivity of the network are consistent with previous implementations

of similar networks [25, 38].

3.4.2 Synaptic plasticity

In [8], synaptic weights evolve with a Spike Time Dependent Plasticity (STDP) rule, according to which long-term synaptic plasticity critically depends on the relative timing of pre- and post-synaptic activity in the millisecond range. In principle, all kinds of synapses should follow specific synaptic plasticity rules, undergoing LTP or long term depression (LTD) according to the local pre- and post-synaptic activity. However, there is not enough experimental evidence to constrain the plasticity rules for all types of synapses used in the network. We have thus initially implemented synaptic plasticity only for the CA3 inputs on CA1, fixing all the other peak synaptic conductances to values consistent with experimental observations. Synaptic plasticity of all synapses will be explored in a future work. Interestingly, the rules for STDP vary widely within brain region, cell, and synapse type [5]. During storage we have applied a STDP learning rule specific for the hippocampus, based on the experimental findings in [87]. Therefore, the induction of LTP or LTD (i.e. the change of the peak synaptic conductance) is determined by the correlation of the pre- and post-synaptic activation as follows:

$$g_{peak}(t) = g^0_{peak} + A(t)$$

with

$$A(t) = A(t-1) \times \left(1 - d \times \frac{e^{-\frac{((t_{post}-t_{pre})-M)^2}{2V^2}}}{V\sqrt{2\pi}} \right)$$

for $(t_{post} - t_{pre}) < 0$ and

$$A(t) = A(t-1) + \left(g^{max}_{peak} - g^0_{peak} - A(t-1)\right) \times p \times e^{-\frac{(t_{post}-t_{pre})}{\tau}}$$

for $(t_{post} - t_{pre}) > 0$.

A is always ≥ 0 and represents the degree of potentiation. The constants $M = -22ms$, $V = 5ms$, $\tau = 10ms$ reproduce the critical time window found in [87]. The g^0_{peak} denotes the initial peak conductance and g^{max}_{peak} its maximum value. The parameters p and d are chosen in such way that, during a time span of $16s$, the potentiated and depressed values for the peak synaptic conductance are consistent with experimental findings [87]. As in [25], the plasticity rule was activated when the local membrane potential crossed a $-55mV$ threshold, consistently with experimental findings suggesting that the post-synaptic neuron did not have to produce an action potential to elicit the plasticity process [32].

Finally, the effect of the putative GABA-B inhibition during the storage phase was modelled by reducing the AMPA component of the CA3 synapses by a 0.4 factor (as in [25]).

3.4.3 Storage and recall

Storage and recall were implemented according to the papers [25, 38, 69]. Inputs from EC and CA3 arrive in CA1 at different phases of a theta cycle. During a half theta cycle, a strong EC input promotes the synaptic modification needed to encode a new pattern, while in the other half theta cycle the CA3 input retrieves old associations without the EC contribution. The storage and recall processes are schematically illustrated in Figure 3.1 **B**, where the complete microcircuit built around a CA1 neuron and the relevant pathways active during each phase are shown. During the storage cycle, the EC inputs activate the (20 randomly selected) CA1 neurons and the iB and iAA interneurons. The CA3 inputs activate iAA, iB, iBS and all the CA1 neurons, with a $9ms$ delay with respect to EC. The iAA and iB generate an inhibitory signal on the CA1 [60], therefore preventing iOLM activation. Note that iB inhibits iBS during the storage phase, so that it is out of phase with the iB and IAA [61] (see the storage phase in Figure 3.1 **B**). If the coincident activity of EC and CA3 on any given CA1 neuron is strong enough, it will be able to potentiate the corresponding synapses, thus storing an object composed by the set of features present in the EC path.

During recall (Fig. 3.1 **B**, right), the septum is active, inhibiting all the inter-neurons. CA1 neurons are relieved from inhibition and activate the iOLM (overcoming septal inhibition). This will in turn inhibit CA1 distal dendrites, preventing any contribution from the EC. The CA3 input will overcome the inhibition only in those CA1 neurons targeted by potentiated synapses, generating spikes in cells that represent the stored pattern. The activation of iBS by the active CA1 neurons will be higher than septal inhibition, and will prevent firing of CA1 cells not belonging to the correct pattern.

3.4.4 Recall quality

The input and output patterns are binary sequences indicating the state of activity (firing +1/quiescent 0) of each neuron. After supplying an input pattern with nP active neurons for storage, the quality of the recall (ranging from 0 to 1) is measured by its correlation with the output pattern, calculated as the normalized dot product:

$$q(P, P') = \frac{\sum_{i=1}^{N} P_i \times P'_i}{\sqrt{\sum_{i=1}^{N} P_i \times \sum_{i=1}^{N} P'_i}}$$

where P is the input pattern, P' the output pattern and N the total number of CA1 neurons.

In general, a higher quality reflects a better recall. More specifically, $q = 1$ indicates that the input and output patterns are the same, while $q = 0$ indicates either that the output has no active CA1 neurons in common with the input, or that all cells are quiescent. A pattern was considered correctly recalled if its recall quality, q, was above the threshold Th. This value corresponds to the quality of recall of an output pattern in which all neurons are active. For this network ($N = 100$, $nP = 20$): $Th = q(P, P') = \frac{20}{\sqrt{20 \times 100}} = 0.4472$. It should be noted that this way to calculate the quality of recall is quite restrictive, since it considers the activity of all the $nP = 20$ neurons belonging to a given pattern, rather than 20 randomly chosen neurons. In the latter case, it will be $q(P, P') = \frac{4}{\sqrt{20 \times 20}} = 0.2$ since there are on average 4 active neurons in common between input and output.

The binary sequence corresponding to the output pattern is formed by considering the CA1 spiking activity during a sliding $5ms$ time window inside the theta hemi-cycle in which an input is presented to the network. Preliminary calculations using a $10ms$ window (as in [25]) did not result in qualitative differences. For each window a binary vector of length 100 is formed, with 0 or 1 according to whether there is at least one CA1 spike within the time window. Each theta recall hemicycle thus contributes with a set of independent values to the average quality. Cases in which there were no spikes over an entire theta recall hemi-cycle were termed "silent cycles".

In [8], we tested two different ways to measure the recall quality, excluding (Q_1) or including (Q_2) silent cycles. Q_1 was defined as the average of the quality values obtained from all time windows with at least 1 spike, i.e silent theta recall hemi-cycles were not considered (as in [25]); Q_2 included silent cycles, which contributed to the average quality with a zero value. Note that these two measures highlight different aspects of the output, Q_1 focusing more on the "quality" of the output patterns, whereas Q_2 is more sensitive to their "quantity". As discussed in details later, we found Q_1 more consistent with experimental findings. We therefore took this as a measure of recall quality in most of the following simulations.

3.5 Biological results

3.5.1 CA1 model

The main aim of work in [8] is to investigate the effects of increased CREB activity on the processes of pattern storage and retrieval in a hippocampal CA1 neuronal network. Experimentally, it has been shown that CA1 neurons become more excitable when higher CREB levels are present [27, 36, 73, 112]. The experimental results obtained in [73], reported in the plot of Figure 3.2 **A**, were chosen as reference data for our CA1 model. These authors explored the role of CREB in modulating the excitability of CA1 neurons by performing single cell recordings in mice that over-express a constitutively active form of CREB (VP16-CREB) in a regulated manner. Among other results, they found that transgene expression increased neuronal excitability (Fig. 3.2 **A**) and inhibited slow and medium potassium currents responsible for after-hyper-polarization currents. We have thus implemented a reduced model of CA1 pyramidal neurons that qualitatively reproduced the experimental findings on the effects of CREB on intrinsic neuronal excitability. The results are shown in Figure 3.2 **A**. We first found the peak channels conductances that qualitatively reproduced the number of action potentials elicited as a function of the somatic current injection under control and, especially, under CREB conditions (Fig. 3.2 **A**). The effect of CREB (Fig. 3.2 **A**, circles) was modeled by decreasing the peak conductance of after-hyper-polarizing currents by the same proportion found in [73], i.e. 52% for mAHP current and 64% for sAHP current. As can be seen, the model results were in good agreement with the experimental findings, reproducing the linear increase of APs with input strength and the roughly 2-fold increase observed for higher CREB levels.

Since we were interested in network properties, we then tested whether the CREB-mediated increase in excitability could also be obtained via synaptic inputs, a condition more similar to in *vivo* activity. Because of the highly non uniform dendritic distribution of ion channels in these neurons (reviewed in [81]), this is important information that cannot be simply inferred from somatic currents injections. We thus carried out additional simulations stimulating the neuron with the synaptic inputs modelling the activation of the CA3 neurons. The results obtained with increasing strengths of synaptic conductance are shown in Figure 3.2 **B** and confirm that, also in this case, the model exhibits the same excitability profiles in control and CREB conditions. These results validate our single neuron model against experimental findings under somatic current injection, and suggest that CA1 neurons excitability increases under CREB, with respect to control, both under a somatic current injection and synaptic inputs.

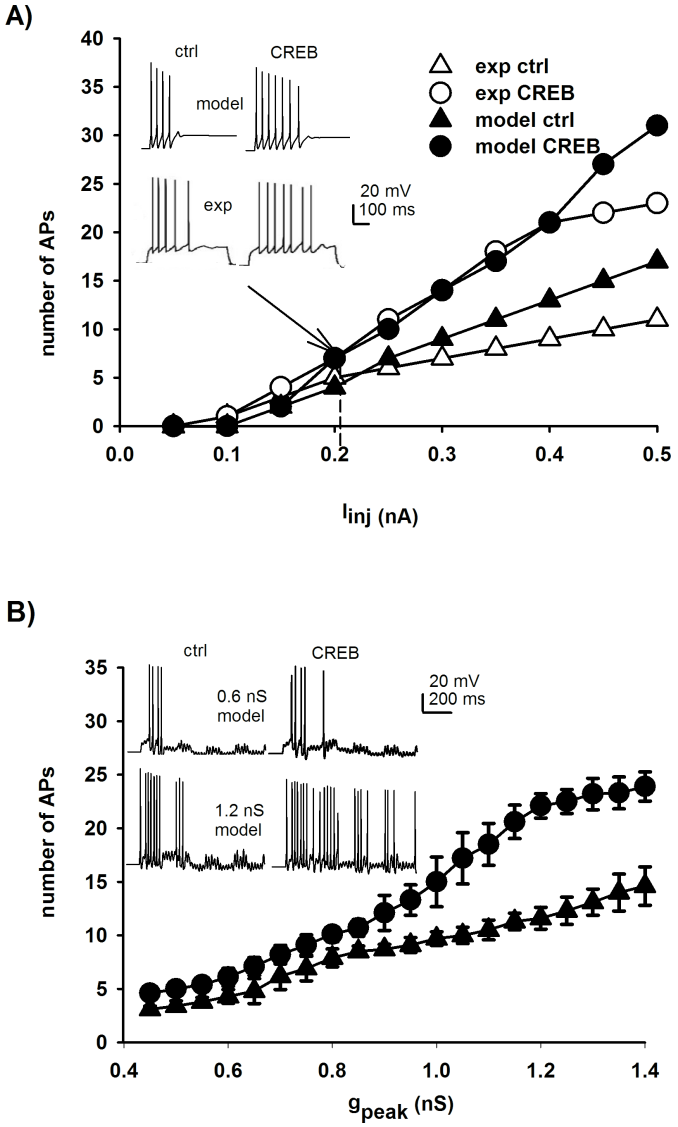

FIGURE 3.2: Input/Output properties of CA1 neurons under control and increased CREB function. **A)** Number of APs triggered in response to 1s depolarizing somatic current injection under control and CREB conditions: experimental findings, taken and redrawn from [73] and results obtained from the model using the same stimulation protocol. **B)** Number of APs elicited in a model neuron during 1s simulations of synaptic activity as a function of the AMPA receptor peak conductance at the CA3-CA1 synapse. Synapses were randomly activated.

3.5.2 Storage and recall of a single pattern

The effect of CREB involves not only the neuron's excitability but also the magnitude of LTP. We had previously studied the consequences of expression of a constitutively active form of CREB on synaptic function in the rodent hippocampus using viral mediated gene transfer in vivo [77]. We had observed that increasing CREB function leads to both an enhancement of NMDA receptor mediated synaptic responses and to an increase in the magnitude of AMPA receptor-mediated LTP, compared to the control neurons. More specifically, the increase in the AMPA receptor mediated Excitatory Post Synaptic

Currents (EPSCs) after LTP induction were larger after CREB activation than in control neurons (by 266% in the CREB case, by 164% in the control case). We therefore also took into account this result in our model. In summary, during a storage cycle we simulate the control case by setting $g_{peak}^{max} = 1.64 g_{peak}^{0}$, and the CREB case by:

- decreasing the peak conductances of mAHP and sAHP currents by 52% and by 64% respectively;
- setting $g_{peak}^{max} = 2.66 g_{peak}^{0}$.

In all simulations, we used $g_{peak}^{0} = 0.45nS$. This value was chosen in such a way as not to have spurious spikes during the random background activity.

The network stores and recalls patterns during the two hemi-cycles of the theta rhythm, intermixing storage and recall [38]. In preliminary simulations, we found that 25 seconds of simulation time, i.e. 100 theta cycles, was enough to allow the synaptic weights to reach an equilibrium value. In Figure 3.3, we show typical results for a few synaptic weights during a simulation. In about $16s$, the conductance peaks reach a maximum value (as in the experimental protocol in [87]) and, for the remaining time, they oscillate within the same range. Although the qualitative time course of the weights is the same under control or CREB conditions, the higher neuronal excitability and LTP magnitude under CREB results in the weights reaching a higher value (Fig. 3.3, right). This effect is robust and was observed for different values of the parameters modulating the plasticity mechanism (not shown). In Figure 3.4, we show typical simulation results during a $1s$ of simulation, after weight equilibration, under control (Fig. 3.4, left) and CREB (Fig. 3.4, right) conditions. The same pattern is recalled on each gamma for all theta cycles. The somatic membrane potential of one of the CA1 neurons belonging to the input pattern (Fig. 3.4, bottom plots) shows that CA1 cells are more active during the recall cycles in the increased CREB case with respect to control. However, in both cases, their spiking activity very closely matches the expected recall, resulting in a very high quality value in all those cases where the CA1 neurons belonging to a pattern generated an AP. This occurs because all the expected 20 CA1 cells generate at least one spike within the theta recall hemi-cycles (in almost all cases). These results suggest that increased CREB activity does not lead to much difference from the control case in terms of the overall quality of the recall (q), although the higher cell excitability may suggest a better robustness of the output during pathological conditions.

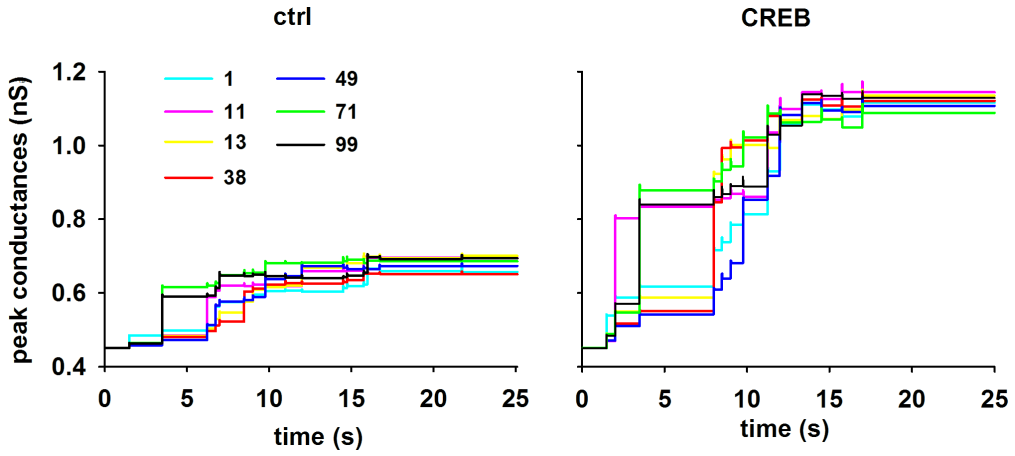

FIGURE 3.3: Plasticity of synaptic weights. Time course for the peak conductance of selected CA3 active synapses (7 out of 100), targeting one of the CA1 neurons involved in a pattern presented during a simulation (100 Θ-cycles of storage/recall); (left) control, (right) under CREB conditions. Synaptic activation times were identical in both conditions. Different colors indicate different CA3-CA1 synapses, targeting the same CA1.

3.5.3 Storage and recall of orthogonal patterns

To better evaluate network performance in storage and retrieval of different patterns, we used 10 groups of 5 orthogonal patterns (i.e. patterns that activate non overlapping synapses), for 5 different stimulation conditions (i.e. we used different random sequences for EC and CA3 inputs activation times). This resulted in 50 sets of 5 orthogonal patterns, for both control and CREB, and corresponded to the maximum number of orthogonal patterns available, given our initial choice to have 20 active CA3 neurons out of 100. To test the storage and recall of these patterns, we carried out a chain of simulations in which a given number of patterns was sequentially stored for 100 theta cycles, one pattern at the time. The recall quality was then tested with a different series of simulations in which the inputs associated with each pattern were presented for 10 theta cycles, with STDP and EC inputs switched off. To allow for equilibration, the first cycle was excluded from the analysis.

In Figure 3.5, we report Q_1 and Q_2 (at the 95% confidence interval) as a function of the stored patterns. For Q_1, the average ($\pm$s.e.m.) overall recall quality for CREB (0.939 ± 0.008, red circles) was slightly but significantly higher than control (0.897 ± 0.002, black triangles, Wilcoxon Signed Rank test $p < 0.001$). These results demonstrate that the recall quality of orthogonal patterns does not significantly change with the number of stored patterns and that increasing CREB activity may only slightly improve an already very good recall quality. For Q_2, the results for the CREB case (Fig. 3.5, green circles) were almost identical to Q_1, because there were very few (1-2) silent cycles. However, Q_2 in the control case (Fig. 3.5, blue triangles) was considerably lower than Q_1, because

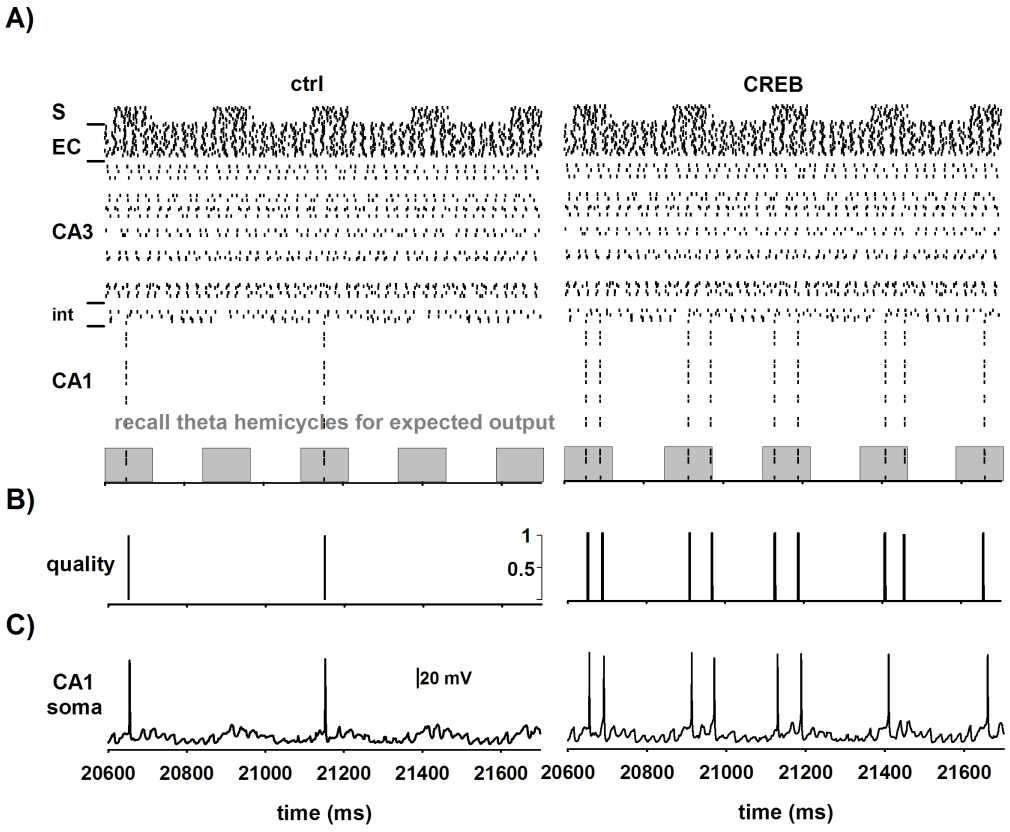

FIGURE 3.4: Typical example of an input/output activity during a simulation. In all cases, left plots refer to control and right plots to CREB. **A)** raster plot showing spike times, during a 1.1s simulation segment, for the septum (10 inputs), EC (20 inputs), CA3 (100 inputs), the 5 inter-neurons, and the 100 CA1 neurons; the gray boxes at the bottom of the plots indicate the time windows for the expected CA1 output (recall phase). **B)** The recall quality for this particular simulation segment. **C)** Somatic membrane voltage for one of the CA1 pyramidal cells receiving both the EC and CA3 inputs.

of the large number of silent cycles (about 5 ± 1 out of 9) that were present under this condition. Contrary to Q_2, Q_1 is representative of the fact that there is no quality improvement by CREB in the healthy case, as demonstrated in [4, 106, 107], in which the same VP16-CREB transgenic mice employed in [73] is studied. We therefore took Q_1 as a measure of recall quality unless otherwise specified.

The results illustrated above represent a healthy condition, in which all synaptic connections evolve according to their activity and the STDP rule. Under these conditions, the recall quality (Q_1) was very high in all cases. We were also interested in investigating the robustness of this result when an increasing fraction of synapses are impaired after the storage phase, as presumably occurs during brain diseases such as AD [98]. To this purpose, we have carried out a set of simulations to calculate the recall quality of a set of 5

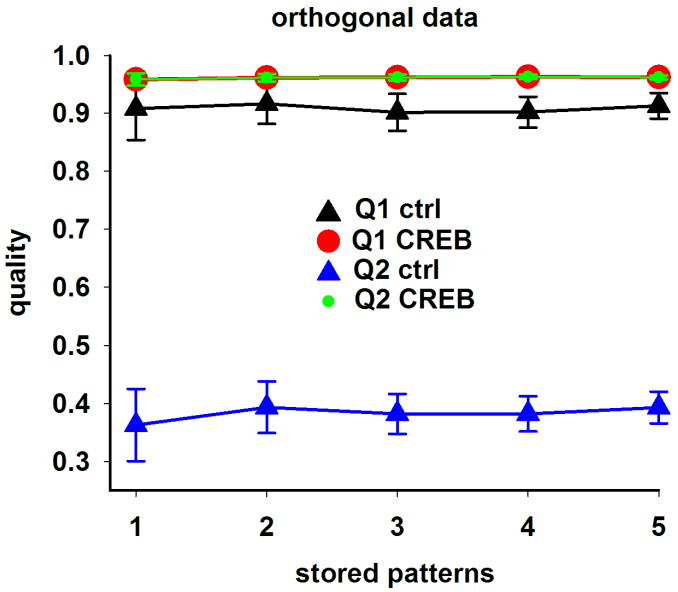

FIGURE 3.5: The recall quality is independent from the number of stored orthogonal patterns. Average recall quality (at the 95% confidence interval) under control (triangles) and CREB (circles) conditions as a function of the number of stored orthogonal patterns with (Q_2) or without silent recall cycles (Q_1) included in the calculation. In all cases, the average values were calculated from the $n \times 50$ values for the recall measure obtained for each number, n, of stored patterns.

orthogonal patterns, after different types and amount of synaptic deterioration. A different proportion of synapses ($20-80\%$ of the whole set of 104 connections) was randomly chosen and their peak conductance reduced by 25 to 75%. The recall quality was then calculated for different combinations of proportion of impaired synapses and amount of peak conductance reduction. The results are shown in Figure 3.6 **A**, where we report the average quality ($\pm$SD, calculated from 10 random groups of involved synapses) as a function of impaired synapses and conductance's reduction of $25-50-75\%$. The 5 patterns were better recalled under increased CREB (Fig. 3.6 **A**, circles) than under control (Fig. 3.6 **A**, triangles) conditions, with a significant increase in the quality for all cases ($19-59\%$ increase, Wilcoxon signed rank test P value < 0.005 in all cases). Remarkably, with increased CREB activity, the quality is always above Th, except in those extreme cases in which 80% of synapses are impaired by a 75% reduction of their peak conductance. Instead, in the control case the quality is below Th in most cases.

We next tested whether one of the mechanisms modulated by CREB had a more important role in determining the recall quality. As discussed in Section Methods, the CREB case was modelled by decreasing the AHP currents and increasing the maximum amount of LTP for CA3-CA1 synapses, with respect to control. In a series of simulations, we analysed the results obtained by modelling the CREB case by changing one condition at the time, as follows:

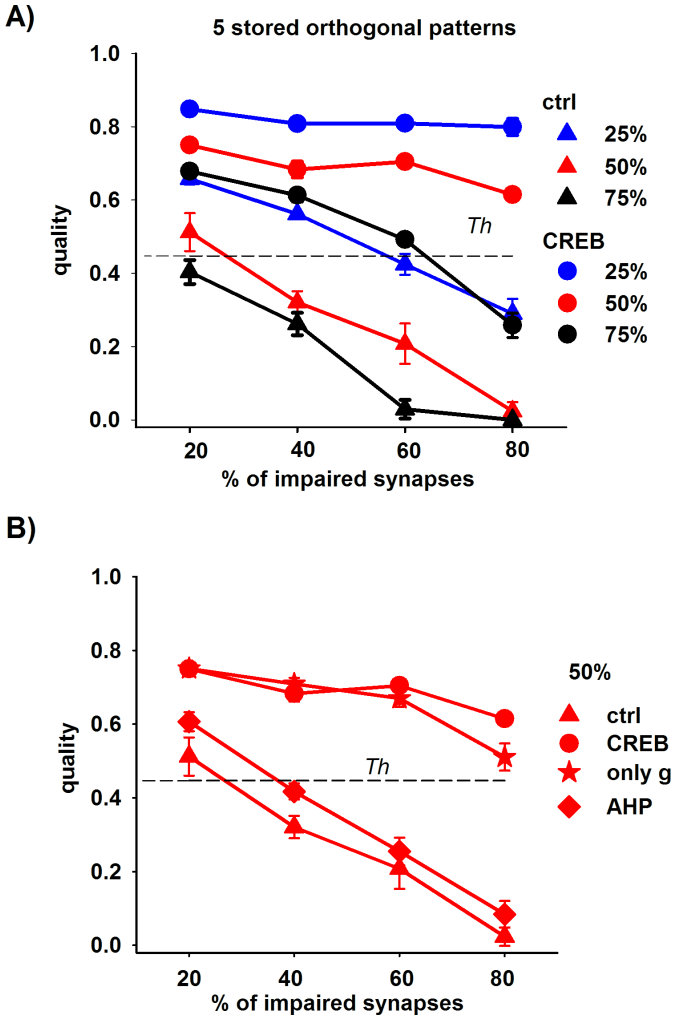

FIGURE 3.6: Increasing CREB function improves the recall quality of orthogonal patterns in a disease-like condition. **A)** Average recall quality under control (triangles) and CREB (circles) conditions as a function of the proportion of impaired synapses, for different peak conductance reductions (25%, 50%, 75%). **B)** Average recall quality as a function of the proportion of impaired synapses after a 50% peak conductance reduction under control (triangles) and CREB conditions modelled as a change in the peak synaptic conductance only (star), decrease in AHP currents only (diamond), and both (circles). In all cases, the average quality was calculated from 10 simulations with random selection of impaired synapses in a representative set of 5 stored orthogonal patterns. The dotted line represents the threshold, Th, for the acceptable quality level.

- case "AHP", where the peak conductances of mAHP and sAHP currents were decreased by 52% and by 64% [73], respectively, and the maximum LTP unchanged with respect to control, i.e. $g_{peak}^{max} = 1.64 g_{peak}^{0}$ [77];
- case "only g", where $g_{peak}^{max} = 2.66 g_{peak}^{0}$ [77] and AHP currents were left unchanged, with respect to control.

The simulation findings for these case are plotted in Figure 3.6 **B**, and show that most of the CREB effects are taken into account by the change in the peak synaptic conductance

(Fig. 3.6 **B**, star). Taken together, these results suggest that an increased CREB activity may significantly improve the storage and recall process under pathological conditions through its effects on synaptic transmission.

3.5.4 Storage and recall of random patterns

Next, we tested the recall quality of an increasing number of random patterns in the healthy condition. This corresponds to an increasing average number of overlapping active neurons (20% for 50 patterns). We have presented 10 different groups of 50 random patterns to the network, for 5 different initial conditions for the firing frequency of EC and CA3 inputs (generating 50 sets of 50 random patterns). Figure 3.7 shows the average quality (Q_1 and Q_2) obtained as a function of the number of stored patterns, using the same storage and recall strategy as for the orthogonal patterns (Fig. 3.5). The recall quality monotonically decreases with the number of stored patterns. The reason for this decrease can be understood by considering that random patterns may have one or more active neurons in common. Thus, each time a new pattern is stored, the synaptic configuration (and in turn the quality of recall) of the previously stored patterns may be altered. In particular, with 20% of active neurons, any two patterns will have, on average, 4 active neurons in common. The storage of each new pattern will thus tend to modify previously active synapses, influencing the recall quality of previously encoded patterns. The effects cumulate each time a new pattern is stored. For example, when the 5-th pattern is stored, the synapses corresponding, on average, to 11.8 active neurons of the first pattern have been modified. This is calculated by using an iterative probabilistic calculation which takes into account the number of overlapping active neurons each time a new pattern is presented. The Q_1 quality for both control and CREB was surprisingly above Th for the entire range, up to 50 patterns, a condition under which all active synapses are involved in more than 25 patterns under physiological conditions (i.e. outside a sparse coding assumption). The neuron's higher excitability and increased LTP under CREB did not significantly change the result obtained in control (Wilcoxon signed rank test $P > 0.1$ in all cases), in agreement with the experimental findings of [4] and [106, 107]. Interestingly, the Q_2 for the control case (blue triangles) was very different from CREB (green circles) and below threshold over the entire range of patterns. This result can be explained with the relatively high mean number of silent cycles included in the calculation (about 5 out of 9 theta hemi-cycles, on average).

To test CREB effects on recall of random patterns under pathological conditions, a fraction of synapses were impaired (i.e. their peak conductance decreased) after the storage phase. We first selected a representative set of 5 patterns, and carried out a series of simulations mimicking synaptic deterioration as discussed in Figure 3.6: for an

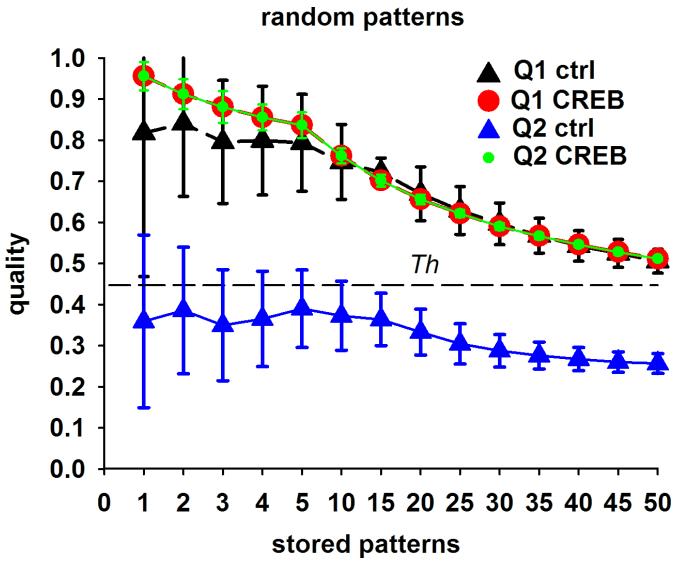

FIGURE 3.7: Increasing CREB function does not improve the recall quality of random patterns in the healthy condition. Average recall quality under control (triangles) and CREB (circles) conditions as a function of the number of stored random patterns with (Q_2) or without silent recall cycles (Q_1) included in the calculation; the dotted line represents the threshold, Th, for the acceptable quality level. In all cases, the average values were calculated from the $n \times 50$ values for the recall measure obtained for each number, n, of stored patterns.

increasing subset of impaired synapses ($20 - 80\%$ of the total), the peak conductance was reduced by different amounts ($25 - 50 - 75\%$) and the average recall quality (Q_1) was calculated in each case. Figure 3.8 **A** displays the results of these simulations. The average quality for the control cases (Fig. 3.8 **A**, triangles) is below Th in almost all cases in which more than $\approx 40\%$ of synapses were impaired, a direct consequence of overlapping inputs. Surprisingly, the control case with a relatively small reduction in the synapses peak conductance (25%, Fig. 3.8 **A** blue triangles), was significantly better than the corresponding case with orthogonal patterns (blue triangles in Fig. 3.6), with the average quality above Th over the entire range tested ($20 - 80\%$ of impaired synapses). This is a case in which the peak conductance of common active synapses is reinforced by the activation of more than one pattern, overcoming the effects of the reduced conductance and leading to a better overall recall quality. Under CREB conditions, the quality was above Th in almost all cases (Fig. 3.8, circles). Only when 80% of synapses were impaired and the peak conductance was reduced by 75%, did the recall quality become unacceptable. Taken together these results suggest that under pathological conditions, random patterns in control (i.e. normal CREB) can be recalled better than orthogonal patterns at the beginning of the disease, whereas CREB activity will significantly improve the recall quality over the entire disease progression.

The robustness of the results for the CREB effects is confirmed by the results shown in Figure 3.8 **B**, where we report the average quality as a function of the stored (random)

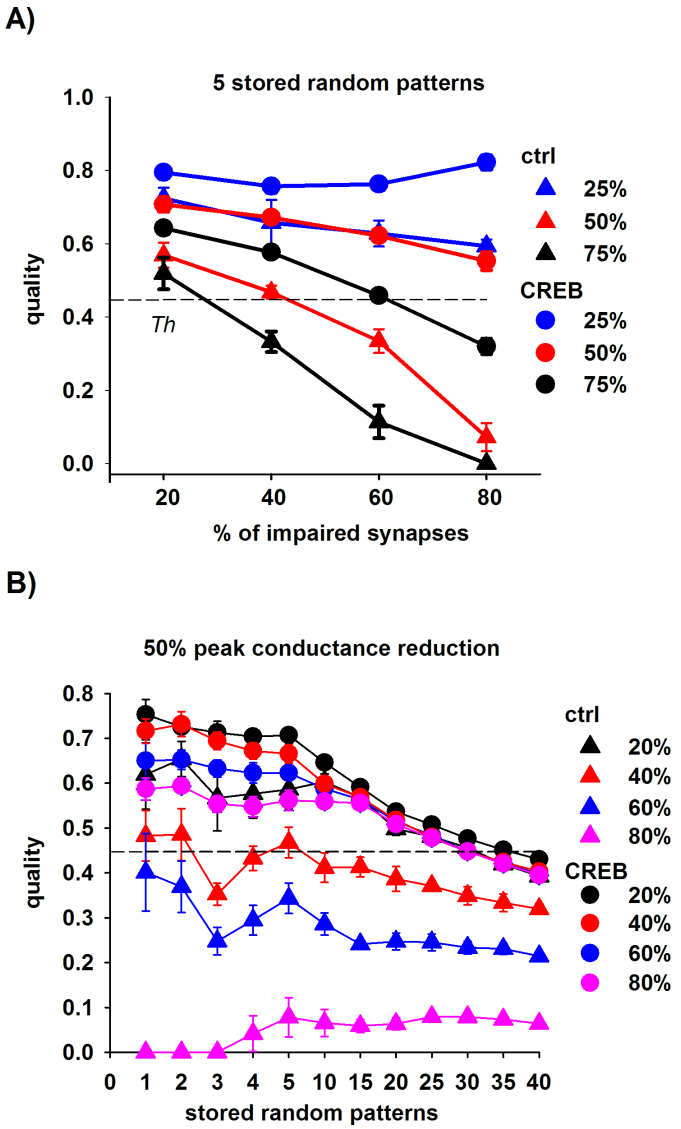

FIGURE 3.8: Increasing CREB function greatly improves the recall quality of random patterns in a disease-like condition. **A)** Average recall quality under control (triangles) and CREB (circles) conditions as a function of the proportion of impaired synapses, for different peak conductance reductions $(25\%, 50\%, 75\%)$. Average quality calculated from the values obtained from 10 simulations with random selection of impaired synapses in a representative set of 5 stored orthogonal patterns. The dotted line represents the threshold, Th, for the acceptable quality level. **B)** Average recall quality under control (triangles) and CREB (circles) conditions as a function of the number of stored random patterns and different proportion of impaired synapses $(20-40-60\%)$. Average values were calculated from the $n \times 50$ values for the recall quality obtained for each number, n, of stored patterns. The dotted line represents the threshold, Th, for the acceptable quality level.

patterns. Assuming a pathology affecting $20-80\%$ of the synapses with a 50% strength's reduction, increasing CREB activity significantly improved the average recall quality with respect to control (Wilcoxon signed rank test $p < 0.002$ in all cases). Particularly striking results were obtained when 80% of the synapses were affected by the reduction (Fig. 3.8 **B**, pink symbols). The poor recall quality under control condition (Fig. 3.8 **B**,

pink triangles) was much improved under CREB (Fig. 3.8 **B**, pink circles) with a quality above threshold for up to 35 stored patterns. These results suggest that increasing CREB activity can greatly improve the process of memory recall under a wide range of pathological conditions.

Finally, we also investigated the effects of CREB activity when synapses are impaired during the storage phase. To this purpose, we carried out a set of simulations using the same group of 50 random patterns used to model a 50% reduction in 20% of the synapses (see Fig. 3.8 **B**). In these simulations, for each group of n patterns, 20% of random CA3-CA1 synapses were impaired during the storage phase of the $n-$th pattern, and the results are shown in Fig. 3.9 **B**. To help the comparison with the results obtained by impairing the synapses before the recall phase (and after storage), in Figure 3.9 **A** we reproduced the quality of n patterns from Figure 3.8 **A** under the same conditions (Fig. 3.9 **A**, recall, solid lines), together with the results obtained by impairing synapses during the storage phase (Fig. 3.9 **A**, storage, dashed lines). Under CREB, the average quality for all patterns did not change (compare circles with solid and dashed lines). Instead, interfering with the storage phase under control conditions had a significant impact, and the overall quality was reduced especially for fewer stored patterns (sparse coding). Interestingly, as shown in Figure 3.9 **B**, under CREB the recall quality of the last pattern (stored with impaired synapses, Figure 3.9 **B**, dashed line and circles) was significantly improved over the entire range of patterns, with respect to the case in which synapses were impaired after storage (solid lines). Under control conditions a better recall was observed only for a higher number of stored patterns (dense coding). These results suggest that CREB activity can maintain an overall better quality with respect to control even when synapses are impaired during the storage phase.

3.6 Discussion

The main aim of this work was to test whether hippocampal single cell properties, altered by increased CREB-dependent transcription, may improve the memory recall process, especially during the progression of a disease such as AD. The ability of a network to store and retrieve information has been under intense scrutiny for the past years. Models of the basic physiological operations of hippocampal networks, as encoding and retrieval processes [70], place fields properties [65], or how inhibitory synaptic parameters can modulate the oscillatory frequency and synchronization properties [110], have been suggested. None of them, however, has implemented and used at the same time multi-compartmental morphologies and synaptic properties to study the effects of increased CREB activation to improve the recall process, as in this work. Several reports have

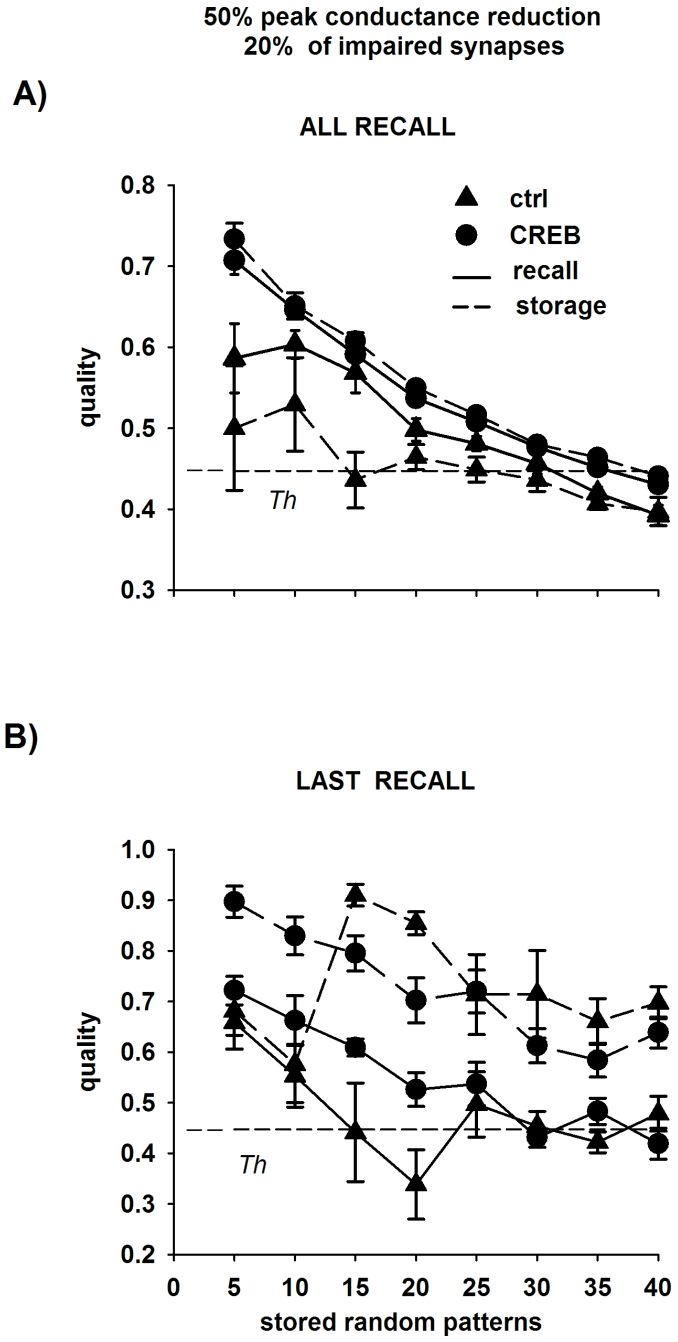

FIGURE 3.9: CREB activity can maintain a better quality when synapses are impaired during the storage phase. **A)** Average recall quality under control (triangles) and CREB (circles) as a function of the number of stored patterns, calculated from simulations in which synapses were impaired after storage of all patterns in each group (recall, solid lines), or before the storage phase of the last pattern (storage, dashed lines). **B)** As in panel A) but with average values calculated from simulations of recall of the last pattern. The dotted line represents the threshold, Th, for the acceptable quality level.

shown that hippocampal CREB signaling is likely to be down-regulated in AD conditions, suggesting negative effects on hippocampal-dependent plasticity [33, 93]. Because of CREB's role in memory formation, the concept of CREB-based memory enhancers, i.e. drugs that would boost memory by CREB activation, has emerged [10, 105]. This

idea is supported by several recent results suggesting that boosting the CREB pathway improves hippocampal-dependent memory in healthy rodents and restores this type of memory in an AD mouse model [2, 33, 85, 96, 97, 111]. However, not much is known about how CREB-dependent neuronal alterations in synaptic strength, excitability, and LTP, which have been observed at the single neuron level using biological tools, can boost memory formation in the complex architecture of a neuronal network. To this purpose, we extended the model in [25] to allow storage/recall of more than one pattern, using a specific STDP learning rule [87] and a CA1 model able to reproduce many experimentally observed features of these neurons under control (e.g. the depolarization block [6]) and under CREB activity [73].

One of the main results of the model is that under healthy control conditions, the larger excitability and LTP magnitude caused by enhanced CREB function does not significantly improve the number of patterns that can be recalled with an acceptable quality. This is true if we assume silent theta cycles are not involved in pattern recall (as in Q_1), using both orthogonal and random patterns. This is a rather surprising result but it is physiologically reasonable for a low excitability neuron like the one we are modelling, consistently with experimental findings on CA1 neurons in control and CREB [73] and with low in-vivo average firing frequencies [44]. That silent cycles may not be relevant for recall can also be alleged by considering the following example: a case in which the relevant CA1 cells are silent in 60% of the theta recall hemi-cycles (poor recall, quality=0) and fire in the rest of the cases (perfect recall, quality=1) would give a meager average quality of 0.4, i.e. a value under threshold for pattern recognition. Furthermore, data are also consistent with experiments suggesting that, while expression of VP16-CREB in transgenic mice facilitates the establishment of hippocampal LTP, it may not improve the recall process, assessed as success in spatial learning. The fact that the relative performance of our network in control and CREB conditions most closely matches spatial learning performance if silent theta cycles are not included in the recall quality calculation, suggests that relatively sparse recall (i.e. every few theta cycles) is sufficient for good cognitive behaviour.

There could thus be additional effects in which CREB activity, through more complex biochemical pathways, could have a more distinct effect on memory-related processes, such as the time of storage/recall or persistence, but a detailed investigation of these effects was out of the scope of this paper. In striking contrast, under AD-like conditions (i.e. with impaired synapses), the model predicts that the effect of CREB over-expression can be remarkable, drastically improving the recall quality even in those cases in which an extensive network impairment would prevent any pattern to be recalled under control conditions. These findings are in good agreement with those of [111], who shown that

locally increasing CREB function in the CA1 region of the dorsal hippocampus is sufficient to rescue the spatial memory deficits of transgenic mice which model key aspects of human AD (poor spatial memory navigation and AD-like neuropathology). The result suggests that the rescue benefits of CREB on memory are mostly mediated by acting on improving synaptic function rather than neuronal excitability, although both are likely to be intimately linked [26]. The model predicts that the use of CREB-based therapies could provide, in principle, a valuable approach to ameliorate AD-related memory deficits, suggesting that pharmacologically stimulating the CREB pathway could be beneficial to rescue memory deficits. There is currently no known therapeutic treatment, which directly increases CREB function. One could however envisage the use of phosphodiesterase IV (PDE IV) inhibitors, such as rolipram, which act upstream of CREB to increase CREB activity. In fact, treatment with this compound was shown to rescue synaptic and cognitive deficits in a mouse model of AD [33]. Translation of our findings to the clinics will however have to await the development of PDE IV inhibitors or other molecules more directly targeting CREB, which are well tolerated in humans [89].

3.7 CA1 microcircuit model

Biological neural networks consist of several kinds of cells interconnected among them. Depending on the biological issues to be studied, it is possible to decide the level of biological details with which to define the neurons, from a computational point of view. Typically, cells representing the inputs of the network are simulated by means of electrical stimuli rather than to be implemented. Conversely, inter-neurons and output cells are biologically defined by means of the implementation of the *neuron's morphology*, *neuron's 3D spatial information* and *neuron's mechanisms* code packages (see Figure 2.1). In the following, we refer to these neurons as *morphological cells*. Figure 3.10 reports the scheme of a generic network model implemented with NEURON, assuming that this consists of K kinds of morphological cells.

As shown in Figure 3.1, the microcircuit consists of $K\_TOT = 8$ different types of cells, for a total of $nCells = 235$ cells. In particular, CA3 ($nCA3 = 100$), EC ($nEC = 20$), and SEP ($nSEP = 10$) cells are the inputs of the network. The last $K = 5$ kinds of neuron, CA1 ($nCA1 = 100$), B ($nB = 2$), BS ($nBS = 1$), AA ($nAA = 1$) and OLM ($nOLM = 1$), are morphological cells obtained by using the standard cell class implementation of NEURON (see Figure 2.2). Figure 3.11 illustrates the main packages of the CA1 microcircuit.

As stated in Section 3.4, the model presented in [8] relies on the neural network described in [25]. This latter is able to perform the steps of storage and recall just for

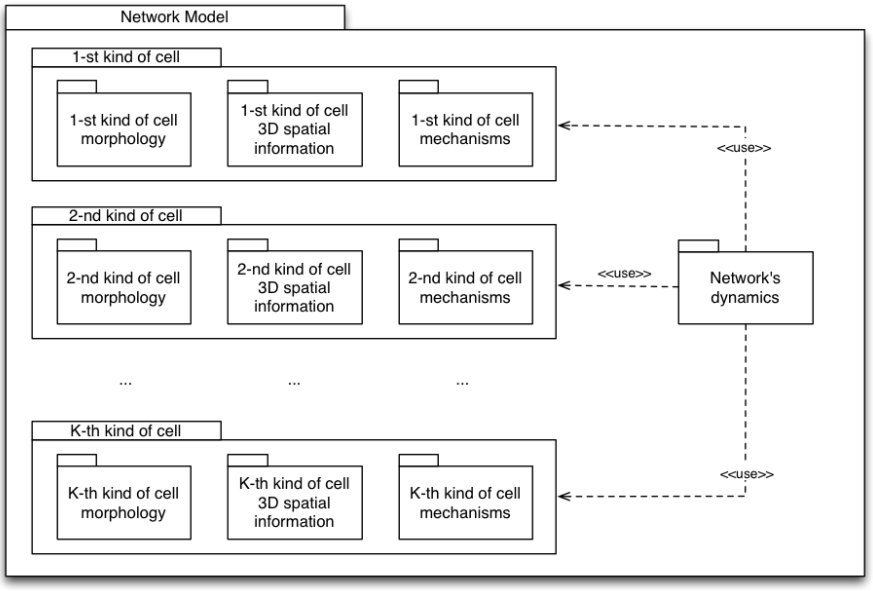

FIGURE 3.10: Model of neural network in NEURON.

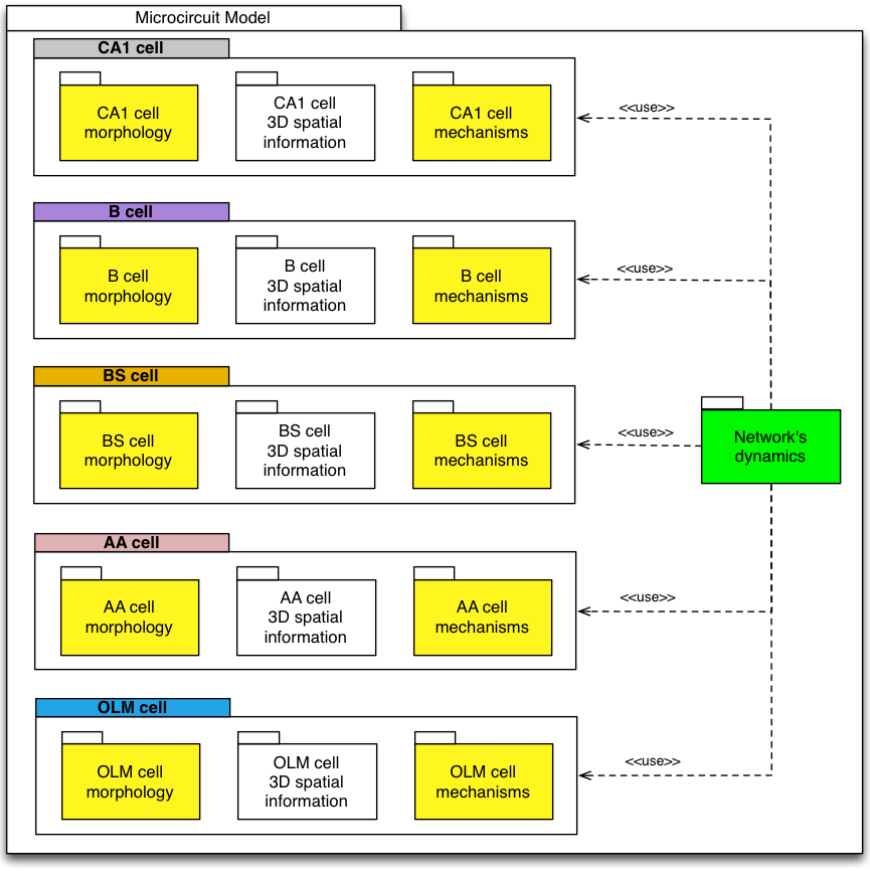

FIGURE 3.11: The CA1 microcircuit model.

a single pattern (see Section 3.4.3). Our modification relies to redesign this model by introducing a new multi-pattern recognition strategy. Hence, given a set of patterns (*n_patterns_to_store* variable), our model is able to store and subsequently to recall all

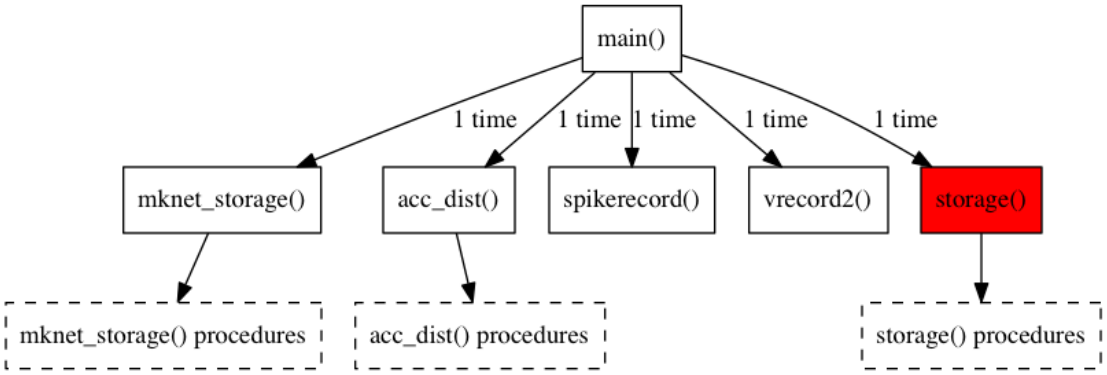

FIGURE 3.12: The computational tree related to the procedure `main()` used in the storage algorithm.

the patterns. As a result, the algorithm is divided in two main sub-algorithms: the storage and the recall schemes.

3.7.1 Storage and recall algorithms

Figure 3.12 shows the computational tree for the storage algorithm, implemented in the *neuron's dynamics* package. This algorithm is organized in two key steps, each of which managed by the procedure `main()`. For each pattern i ($1 \leq i \leq n\_patterns\_to\_store$) to be stored, let be

$$\mathbf{W}^{(i)} = s(\mathbf{W}^{(i-1)}, \mathbf{c}^{(i)}, \mathbf{p}^{(i)})$$

where $\mathbf{W}^{(i-1)}$ is the weight matrix obtained from the storage of the $i-1$-th pattern ($\mathbf{W}^{(0)} = 0$), $\mathbf{c}^{(i)}$ is a network connection vector, $\mathbf{p}^{(i)}$ is the i-th pattern to be stored, and s is a function that represents the STDP rule. The weight matrix $\mathbf{W}$ covers a key role in the proper setting of the network connections.

More in detail, the first step is the network creation, which consists in the set up of the cells, and in the placement of the synapses by means of $\mathbf{W}$. This task is carried out, only one time, by resorting to the procedures `mknet_storage()` and `acc_dist()`. In Figure 3.13 the computational tree related to `mknet_storage()` is drawn. This calls `alzh()`, which simulates Alzheimer's disease by modifying the initial weight matrix $\mathbf{W}^{(0)}$, according to the decay percentage of synapses. Moreover, `mknet_storage()` calls the procedures `mkcells()` and `mkinputs()`, which create and set up the morphological and the input cells, respectively. Finally, the procedures `connectcells()`, `connectEC()`, `connectEC_2()` and `connectCA3()` are called for placing the synapses among the cells and for configuring them by using $\mathbf{W}^{(0)}$.

The second step is referred to the numerical integration of the model, where $\mathbf{W}^{(i)}$ is updated for each i-th pattern to be stored. In detail, `main()` calls the procedures

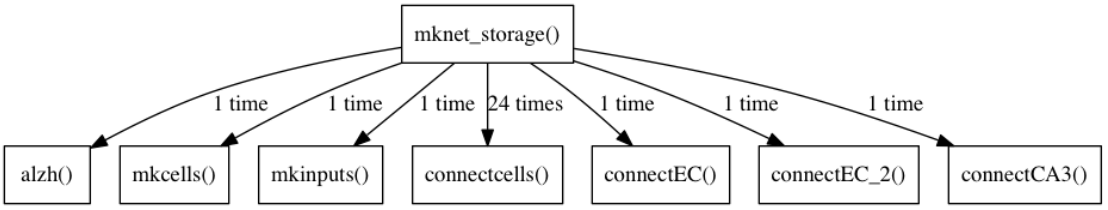

FIGURE 3.13: The computational tree related to the procedure `mknet_storage()` used in the storage algorithm.

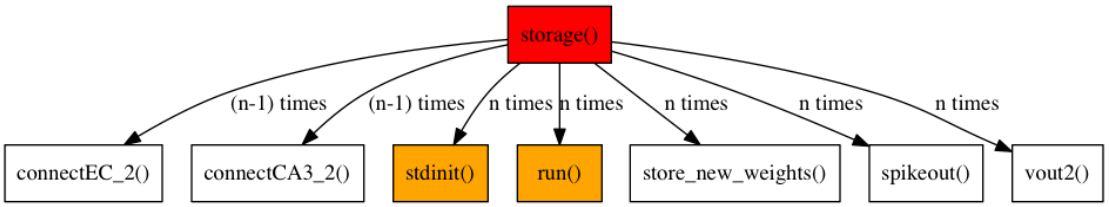

FIGURE 3.14: The computational tree related to the procedure `storage()` used in the storage algorithm.

`spikerecord()` and `vrecord2()`, which record the firing times and the potential values for each cell, respectively. Then, the storage of the patterns is completely performed by `storage()` (in the red box), which iteratively calls some sub-procedures, for each pattern to be stored. In the following we give a brief description of the kernel procedure `storage()`, which is very computationally expensive. As we can see in Figure 3.14, where for simplicity we renamed *n_patterns_to_store* with n, the procedures `connectEC_2()` and `connectCA3_2()` are called. For each pattern i (with $i > 1$), these are delegated to change the network connections by using $\mathbf{W}^{(i-1)}$, which has been updated after the storage of the previous pattern. Then, the NEURON's standard procedures (in the orange boxes) `stdinit()`, which initialize the model, and `run()`, which integrates the model, are invoked. Moreover, the procedure `store_new_weights()` is used to update $\mathbf{W}^{(i)}$. Finally, the procedures `spikeout()` and `vout2()`, which produce the output information previously recorded, are called.

Figure 3.15 shows the computational tree for the recall algorithm, implemented in the *neuron's dynamics* code package. The scheme for the recall of a set of patterns (*n_patterns_to_recall* variable) is organized in two key steps, each of which managed by the procedure `main()`, as well as for the storage algorithm. Let be

$$OUT^{(i)} = r(\mathbf{W}^{(N)}, \mathbf{c}^{(i)}, \mathbf{p}^{(i)})$$

where $\mathbf{W}^{(n)}$ is the weight matrix obtained from the patterns previously stored, $\mathbf{c}^{(i)}$ is a network connection vector, $\mathbf{p}^{(i)}$ is the i-th pattern to be recalled. The function r evaluates the recall and gives in output $OUT^{(i)}$, which is the biological response related to the recall quality.

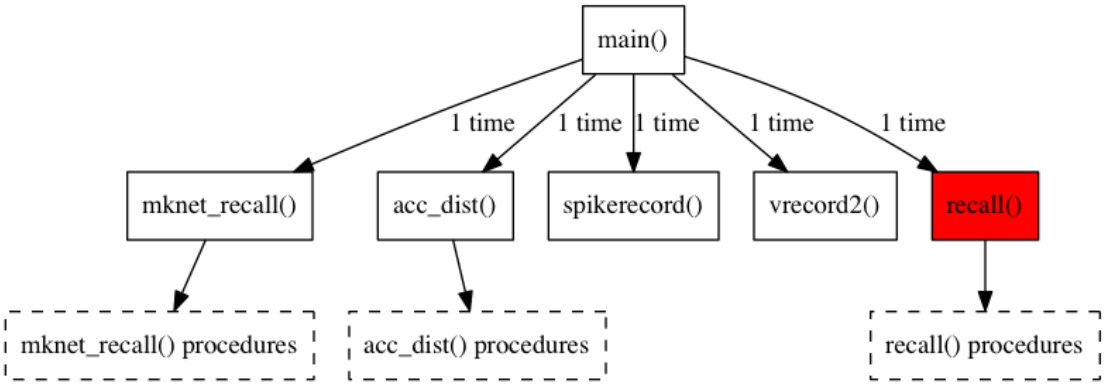

FIGURE 3.15: The computational tree related to the procedure `main()` used in the recall algorithm.

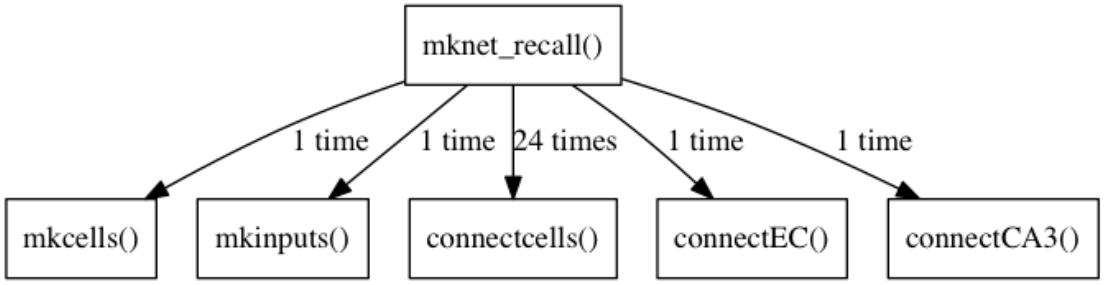

FIGURE 3.16: The computational tree related to the procedure `mknet_recall()` used in the recall algorithm.

The first step of the recall algorithm is the network creation, as well as for the storage algorithm. In Figure 3.16 the computational tree of the procedure `mknet_recall()` is drawn. Notice that this differs from the storage version because it does not call the procedures `alzh()` and `connectEC_2()`.

The second step consists in the numerical integration of the model, performed by the procedure `recall()` (in the red box), which iteratively, for each pattern to be recalled, calls some sub-procedures. This procedure represents the most expensive part of the recall algorithm. As we can observe in Figure 3.17, where for simplicity we renamed *n_patterns_to_recall* with n, `recall()` calls the NEURON's standard procedures `stdinit()` and `run()` (in the orange boxes), and then it resorts to the procedures `spikeout()` and `vout2()`.

3.8 Parallel tools for CA1 microcircuit model

As stated in Section 2.5, the tuning phase of a complex model of single neuron requires several thousand of simulations in order to properly reproduce biological behaviours. For our CA1 microcircuit this issue emerges even more, because the number of biological

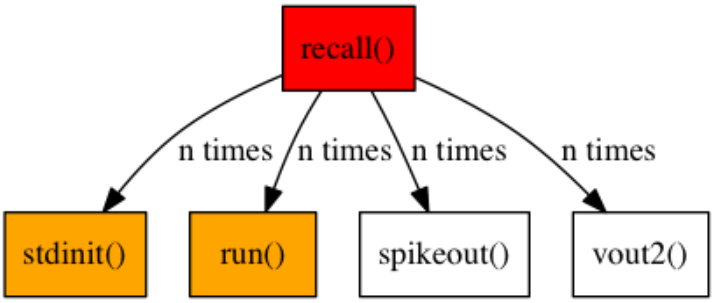

FIGURE 3.17: The computational tree related to the procedure `recall()` used in the recall algorithm.

parameters and synaptic mechanisms to be tuned grows with the number K of the different types of morphological neurons ($K = 5$) in the network. In fact, each kind of cell adds to the simulation runtime a different amount of processing time. This is due to the number of compartments, the types of channels in a single compartment, and the complexity and the number of synapses [42]. As a result, it is needed to resort to parallel techniques, in order to perform simulations with reasonable execution times. NEURON is able to support the parallelization of network models while maintaining, as much as possible, a separation between the network features and how the cells are distributed among all the processors. Hence, without modification, it is possible to write code that can be properly executed in any serial or parallel hardware environment, and that produces quantitatively identical results regardless of the number of CPUs [41]. More in detail, we start from the *network's dynamic* code package shown in Figure 3.10 (in the green box), which reproduces the experiments on the CA1 microcircuit. Then, for each kind of cell, we properly set up the parameters of the *neuron's morphology* and the mechanisms of the *neuron's mechanisms* packages (in the yellow boxes), without changing the *neuron's 3D spatial information* package.

Algorithm 2 The round robin algorithm.

```
1: for (gid = pc.id; gid < nCells; gid += pc.nhost) {
2:   //create cell associated with gid
3: }
```

The parallelization of a neural network model in NEURON consists of a mapping of the network cells to the available processors (i.e., cores). Obviously, it is necessary to introduce an integer global identifier (`gid`) in order to identify any cell. The distribution of the gids (i.e., cells) on the processor is a crucial step in terms of performance. The ideal condition is that each processor has approximatively the same amount of sequential work to do. The simplest load balance approach ignores differences in individual cell processing [42], according to the round robin (RR) strategy shown in Algorithm 2.

Host ID	GID	LID
0	0, N, 2N, ...	0, 1, 2, ...
1	1, N+1, 2N+1, ...	0, 1, 2, ...
⋮	⋮	⋮
N-1	N-1, 2N-1, 3N-1, ...	0, 1, 2, ...

TABLE 3.1: Mapping processors-cells with round robin strategy. Processor ID: processor identifier; GID: global identifier of the cell; LID: local identifier of the cell on a specific processor.

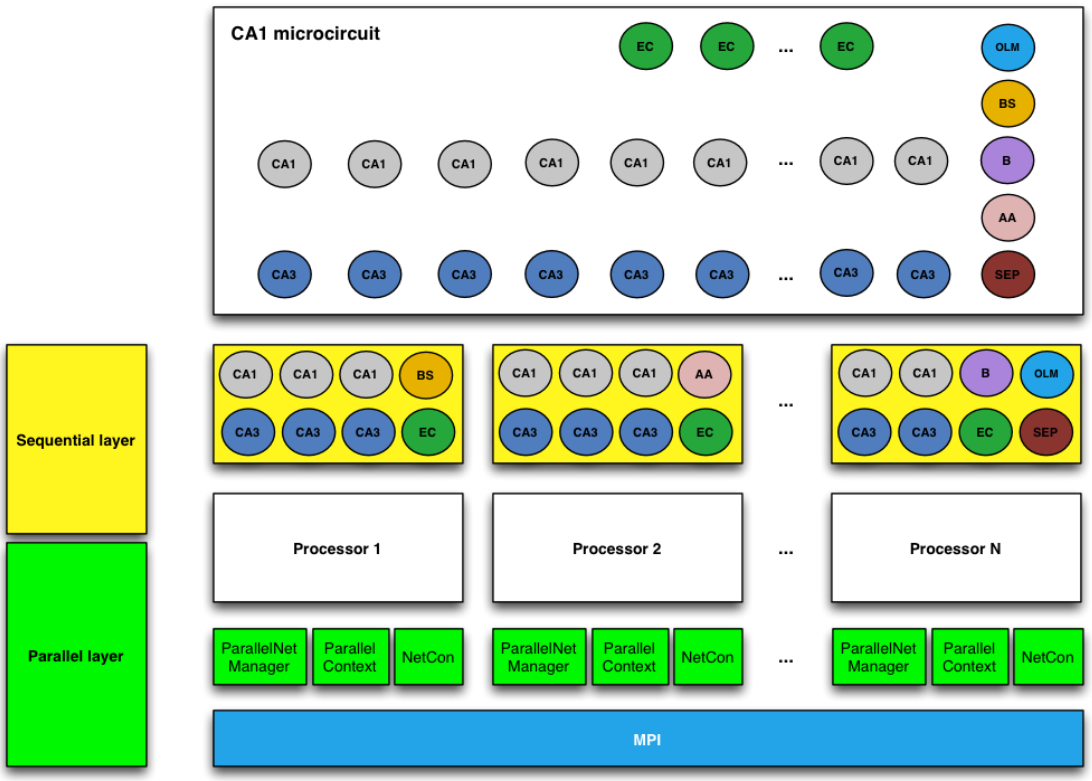

FIGURE 3.18: Parallel framework for the CA1 microcircuit.

Here, `pc.nhost` is the number of available processors, `nCells` is the number of cells and `pc.id` identifies the i-th processor. In Table 3.1, assuming that the number of available processors is $N =$ `pc.nhost`, the mapping of cells to processors with the RR algorithm is shown.

Adopting the RR strategy, we introduced a parallel layer in the *network's dynamics* package. Here, as we can see in Figure 3.18, the underline communication tool is MPI (in the blue box), as well as for the CA1 model [6] described in Chapter 2. The yellow boxes represent the sequential part of the code, related to the groups of neurons assigned to the same processor (in the orange box). The set up of a parallel network uses the `NetCon` class as much as possible [84]. Moreover, additional parallel specific NEURON's methods from the classes `ParallelContext` and `ParallelNetManager` are required. For

more details on these three classes (in the green boxes) see Figure 2.12. In this way, there is no *master* host and all processors perform exactly the same program on different subsets of neurons. All together the cores *set up* the internal and external connectivity and perform the *numerical integration* on its associated cells. In the next subsections we will present some parallel implementation of the storage and the recall algorithms.

3.8.1 Data consistency

One of the key issues addressed to parallelize an application is the data consistency: each processor can only access data that actually handles. For the storage and the recall algorithms this means that there is no processor attempting to manipulate cells that it does not handle. Hence, it is very important to properly manage the communication of a spike between two cells assigned to different processors. NEURON uses an event delivery system to implement spike-triggered synaptic transmission among cells. In the simplest case on serial hardware, the connection between a spike source (pre-synaptic cell) and its target is made by instantiating an object of the class `NetCon` executing a statement of the form

```
nc = new NetCon(source, target)
```

to monitor a source (pre-synaptic cell) for spikes. The detection of a spike throws an event which will be delivered to the `NetCon`'s target [41]. In a parallel simulation environment, the solution is to give to any cell (spike source) its own `gid` that can be referred by any processor.

The aforementioned aspects are managed by the parallel implementations of the procedures for cell connection `connectcells()`, `connectEC()`, and `connectCA3()`. In Algorithm 3, the procedure `P_connectcells()` is shown. Here, the processor connects the target cells `target`, belonging to it, to the source cell `source` by means of `gid_connect()` NEURON's method (line 14). This procedure is recalled by `mknet_storage()` (see Figure 3.13) and `mknet_recall()` (see Figure 3.16), for the storage and recall respectively.

More in detail, Algorithm 4 shows the parallel implementation of `mknet_storage()`, renamed as `P_mknet_storage()`, while Algorithm 5 lists the parallel implementation of the procedure `mknet_recall()`, renamed as `P_mknet_recall()`. In Algorithms 6 and 7 the parallel implementations of `main()` and `storage()` procedures are illustrated (Figures 3.12 and 3.14). Finally, in Algorithms 8 and 9 the parallel implementations of `main()` and `recall()` procedures are shown (Figures 3.15 and 3.17).

Algorithm 3 A light view of the `P_connectcells()` procedure.

```
1: proc P_connectcells() {
2:   ...
3:   // loop over possible target cells
4:   for i=0, nCells-1 {
5:    gid = gidvec.x[i] // global id of cell
6:    if (gid >= first_target_cell_index && gid < last_target_cell_index)
   {
7:     ...
8:     // return in r the source cell index
9:     rs.r.discunif(first_source_cell_index, last_source_cell_index)
10:    source = rs.repick()
11:    ...
12:    // set up connection from source to target
13:    target = cells.object(i).pre_list
14:    nc = pc.gid_connect(source, target)
15:    ...
16:   }
17:   ...
18:  }
19:  ...
20: }
```

Algorithm 4 A light view of the `P_mknet_storage()` procedure.

```
1: proc P_mknet_storage() {
2:   ...
3:   /*** Alzheimer application:  synapse decay ***/
4:   if (pc.id==0) P_alzh(alzheimer_percentage)
5:   pc.barrier()
6:   /*** Cell creation ***/
7:   P_mkcells() // create the morphological cells
8:   P_mkinputs() // create CA3, EC and SEP inputs
9:   ...
10:  /*** Network connection based on the weight matrix W^(0) ***/
11:  P_connectcells(nB, nEC)
12:  ...
13:  P_connectcells(nB, nOLM)
14:  P_connectEC()
15:  P_connectEC_2(pattern, 1, n_patterns_to_store) // Store the 1-st
   pattern...
16:  P_connectCA3(alzheimer_weight_matrix, 0, connection_probability) //
   ...with modifiable synapses
17: }
```

3.9 Performance results

In this Section we report the performance results related to the described parallel algorithms. Overall simulations are performed with NEURON (version 7.1), by using the

Algorithm 5 A light view of the `P_mknet_recall()` procedure.

```
1: proc P_mknet_recall() {
2:   /*** Cell creation ***/
3:   P_mkcells() // create the morphological cells
4:   P_mkinputs() // create CA3, EC and SEP inputs
5:   ...
6:   /*** Network connection based on the weight matrix W^(0) ***/
7:   P_connectcells(nB, nEC)
8:   ...
9:   P_connectcells(nB, nOLM)
10:  P_connectEC() // Restore existing pattern...
11:  P_connectCA3(weight_matrix, 0, connection_probability) // ...with
   modifiable synapses
12: }
```

Algorithm 6 A light view of the `P_main()` procedure for the storage algorithm.

```
1: proc P_main() {
2:   ...
3:   /*** Network creation and connection based on the weight matrix
   W^(0) ***/
4:   P_mknet_storage(weights_matrix, 0, connection_probability)
5:   P_acc_dist()
6:   /*** Instrumentation ***/
7:   P_spikerecord()
8:   P_vrecord2()
9:   ...
10:  /*** Numerical integration ***/
11:  P_storage()
12:  ...
13: }
```

model files available on ModelDB (`http://senselab.med.yale.edu/ModelDB/`), with identification number 151126 [9], and they were run with MPI using up to 128 cores on the S.Co.P.E. Grid infrastructure at University of Naples Federico II, Naples, Italy. For more details on the infrastructure used, see Section 2.6. The storage and recall algorithms were executed on a set of 10 patterns, with an initial delay of $100ms$ and 8 theta-cycles of $250ms$. Hence, the overall simulation time is $100ms + (250ms \times 8) = 2100ms$. Notice that real experiments were performed with 50 patterns and 100 theta-cycles, for an overall simulation time equal to $100ms + (250ms \times 100) = 25.1s$.

In order to evaluate the storage and recall performance, different parameters were taken into account. More in detail, **Runtime** indicates the total execution time, **Wait** represents the time spent for exchanging spikes during a simulation, and **Step** is the time spent for numerical integration. Moreover, **Setup and cell creation** represents the time needed to create the cells, **Network connection** indicate the time for connecting

Algorithm 7 A light view of the `P_storage()` procedure.

```
proc P_storage() {
  for i = 1, n_patterns_to_store {
   if (i != 1) {
    /*** New network connection based on the weight matrix W^(i-1)
***/
    P_connectEC_2(pattern, i, n_patterns_to_store) // Store the i-th
pattern...
    P_connectCA3_2(weights_matrix, i-1) // ...with modifiable synapses

   }
   ...
   /*** Numerical integration ***/
   pc.set_maxstep(1)
   stdinit()
   pc.psolve(tstop)
   /*** New weight matrix W^(i) storage ***/
   P_store_new_weights(weights_matrix, i)
   ...
   /*** Output generation ***/
   P_spikeout()
   P_vout2()
   ...
}}
```

Algorithm 8 A light view of the `P_main()` procedure for the recall algorithm.

```
proc P_main() {
  ...
  /*** Network creation and connection based on the weight matrix
W^(0) ***/
  P_mknet_recall(weights_matrix, 0, connection_probability)
  P_acc_dist()
  /*** Instrumentation ***/
  P_spikerecord()
  P_vrecord2()
  ...
  /*** Numerical integration ***/
  P_recall()
  ...
}
```

the network, and **New weight matrix storage** is the time for collecting and storing the values of the weight matrix. Finally, **Output** is the time for storing biological output information, and **Others** represents the time for collecting non-functional information.

Figure 3.19 shows that the CA1 microcircuit well scales on a single server (node) of the S.Co.P.E. infrastructure. In fact, moving from 1 up to 8 processors, we observe a huge reduction of **Runtime** from 40934s to 3189s. Moreover, Figure 3.20 shows that

Algorithm 9 A light view of the `P_recall()` procedure.

```
1: proc P_recall() {
2:   for i = 1, n_pattern_to_recall {
3:     ...
4:     /*** Numerical integration ***/
5:     pc.set_maxstep(1)
6:     stdinit()
7:     pc.psolve(tstop)
8:     /*** Output generation ***/
9:     P_spikeout()
10:    P_vout2()
11:    ...
12: }}
```

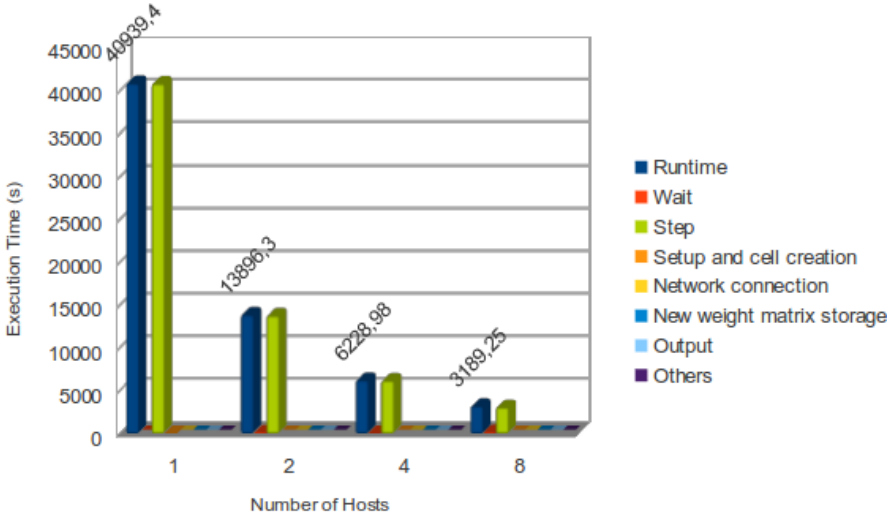

FIGURE 3.19: Execution times for the storage of 10 patterns with 8 theta cycles moving from 1 to 8 processors.

the numerical integration time (i.e., **Step**) dominates the overall execution time. More in detail, with 8 cores the **Step** time is the $\sim$ 95% of the **Runtime**, while the **Wait** time is the $\sim$ 4%. Finally, the other execution steps (setup and cell creation, network connection, weight matrix storage and output generation) do not affect the overall time.

Figure 3.21 illustrates an high worsening of the performance, moving from 8 to 128 processors. More in detail, despite the numerical integration time (**Step**) continues to decrease with doubling of the core number, the **Runtime** remains almost constant, although with slight deteriorations. In fact, as we can observe in Figure 3.22, **Step** ranges from $\sim$ 95% with 8 cores to 6% with 128 cores. On the other hand, there is a strong increment of the amount of time needed to synchronize the communication between two cells mapped on two different processors (**Wait**). Indeed, moving from 8 to 16 cores, we note a huge increasing from $\sim$ 4% to $\sim$ 48%, until reaching the value $\sim$ 87% with 128 processors. This phenomenon is due to the communications. In general,

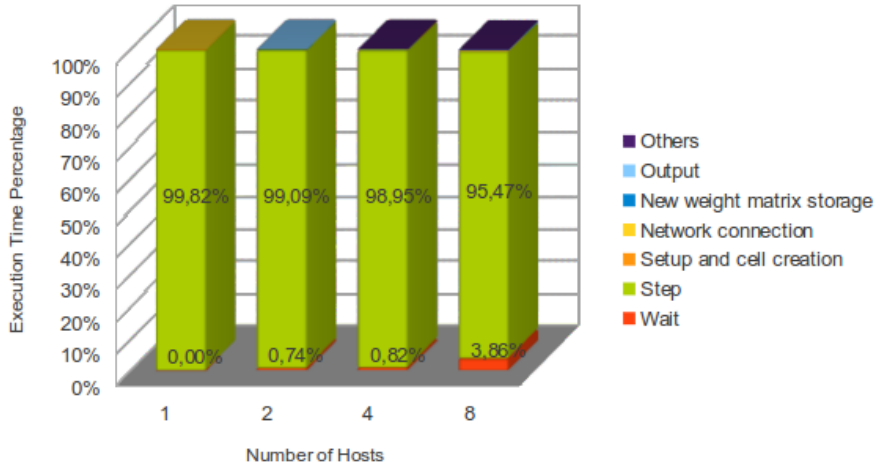

FIGURE 3.20: Percentages of execution time steps for the storage of 10 patterns with 8 theta cycles moving from 1 to 8 processors.

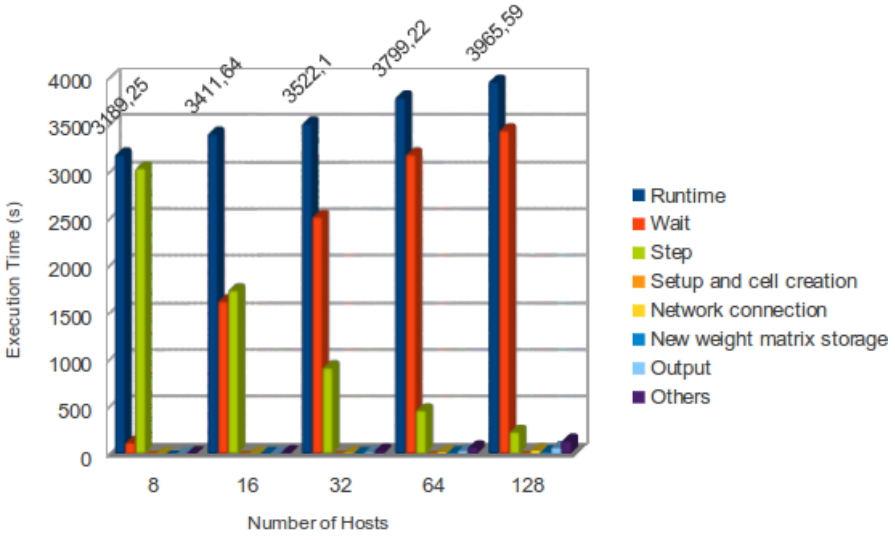

FIGURE 3.21: Execution times for the storage of 10 patterns with 8 theta cycles moving from 8 to 128 processors.

increasing the number of processors, the size of subnets assigned among the processors becomes small and the communication overhead for MPI calls larger. In these conditions, communication times begins to dominate the runtime [84].

The recall algorithm has the same performance we have discussed for the storage. But, in this case, it is possible to implement the recall algorithm on a distributed architecture: in this way, each algorithm performs the recall of a single pattern on a node of the infrastructure. A typical parametric execution of a distributed recall phase is characterized by the pair (*n_patterns_to_recall*, *n_theta_cycles*). In Table 3.2 the performance of the parametric execution (10 patterns, 16 theta cycles) are illustrated, for an overall simulation time equal to $100ms + (250ms \times 16) = 4100ms$ for each pattern. The recall of *n_patterns_to_recall* patterns is performed in $\sim 600s$, which is the time needed to recall

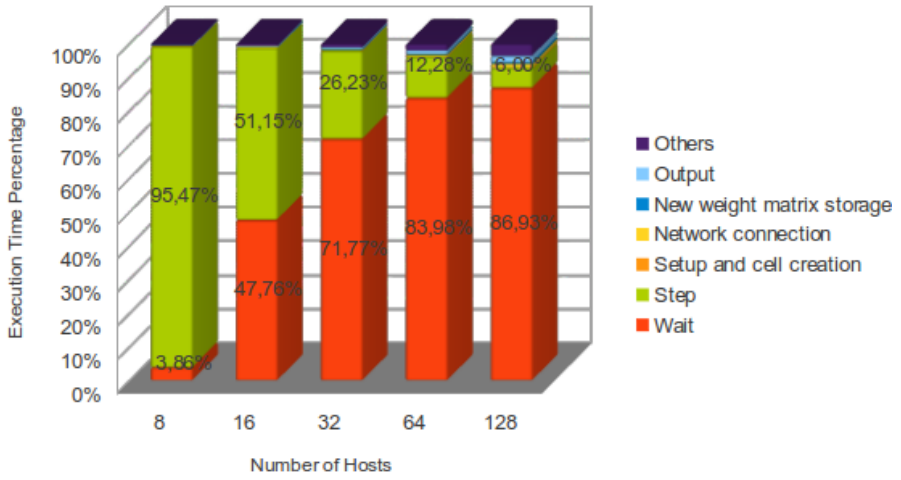

FIGURE 3.22: Percentages of execution time steps for the storage of 10 patterns with 8 theta cycles moving from 8 to 128 processors.

just one single pattern, obtaining a substantial performance improvement: without the distribution of the patterns, the **Runtime** would have been equal to $\sim 6000s$.

Pattern ID	Runtime	Wait	Step	#Spikes	Set and Connect	Output
1	608,775	8,71033	597,987	7975	0,54125	0,61625
2	608,861	8,93715	598,457	8149	0,54125	0,6375
3	607,072	8,51312	597,083	8061	0,54125	0,64875
4	607,812	8,67061	597,683	8033	0,54125	0,635
5	607,119	9,37756	596,286	7946	0,54125	0,6275
6	607,035	8,71243	596,895	7992	0,54125	0,6325
7	607,726	8,79549	597,473	8030	0,54125	0,62125
8	607,191	8,39568	597,321	7848	0,54125	0,63875
9	608,27	10,0982	596,692	8022	0,54125	0,63625
10	608,42	9,3668	597,576	8046	0,54125	0,6325

TABLE 3.2: Execution times with 8 cores for the recall of 10 patterns with 16 theta cycles, distributing the patterns.

Finally, we remark that there exist alternative strategies to the RR, which allow to obtain a better load balancing. One of these is the "longest processing time" (LPT) algorithm in which one iteratively chooses the largest cell and puts it on the currently lest used processor [42]. In this case, the METIS [57] graph partitioning program can be used to define a `ngid` on `nhost` partition for optimizing the load balance and for minimizing the communications [84]. We remark that we have chosen to resort to the simplest strategy of distribution of the cells (i.e., RR) for the following discussion. As observed in [84], in a context where computation time no longer scales with increasing number of CPUs, we have a large number of spikes with constant synaptic delay. In our model, there are the 55% of fast spiking artificial cells (input cells), which make this point dramatically by communications point of view.

3.10 Conclusions

The simulation of biological neural networks is a challenging application from a computational point of view. The calibration and set up of a network require mathematical models in order to simulate the biological behaviour of the different cell type and sophisticated programming environments for developing simulation codes. In practice, building up a microcircuit that mimes the real behaviour of a biological neural network, with a large number of connections between its neurons, requires algorithms and communication strategies computationally expensive. We observed that the main problem that has to be overcome is the communication between a source cell and its target, in a parallel simulation environment. To overcome this problem, we strongly suggest to use general-purpose simulation environments that support massively parallel multi-core programming. Finally, we think that the performance analysis of the proposed microcircuit is useful in order to simulate a large number of microbiological multi-pattern recognition experiments in an acceptable computing time.

Chapter 4

Biologically inspired models describing user behaviours in a Cultural Heritage scenario and a social network community

Abstract. *In this Chapter we start proposing a biologically inspired mathematical model to simulate the personalized interactions of users with cultural heritage objects. The main idea is to measure the interests of a spectator with respect to an artwork by means of a model able to describe the behaviour dynamics. In our approach, the user is assimilated to a computational neuron, and its interests are deduced by counting potential spike trains, generated by external currents. The key idea consists in comparing a strengthened validation approach for neural networks based on classification with our novel proposal based on clustering; indeed, clustering allows to discover natural groups in the data, which are used to verify the neuronal response and to tune the computational model. Moreover, a biologically inspired mathematical model to simulate social network behaviours is presented. We propose a computational network of Integrate & Fire neurons to model the dynamics of spectators in a community that shares interests on cultural heritage assets. In this approach, users are assimilated to neurons, while the friendships and the common interests are the synapses (i.e., connections) among them. These connections may be more or less strong in dependence of how interests to a particular feature are shared. The main novelty consists in describing the propagation of the information on the network by simulating various applicative scenarios.*

4.1 Main references

[22] **De Michele P.** et al.: *A biologically inspired model for describing the user behaviors in a Cultural Heritage environment.*

Abstract. *We present a biologically inspired mathematical model for describing the personalized interactions of users with cultural heritage objects. The main idea is to measure the interest of an artwork spectator by means of a model able to capture the context and the visitor behaviors. In the proposed approach, the user is assimilated to a computational neuron, and its interests are deduced by counting potential spike trains, generated by external currents. Preliminary experimental results based on a phantom database, built from a real scenario, are shown. Finally, comparison results and application scenarios that may be adopted for the context-aware user profiling in the cultural heritage framework are discussed.*

[24] **De Michele P.** et al.: *A clustering-based approach for a finest biological model generation describing visitor behaviours in a Cultural Heritage scenario.*

Abstract. *We propose a biologically inspired mathematical model to simulate the personalized interactions of users with cultural heritage objects. The main idea is to measure the interests of a spectator with respect to an artwork by means of a model able to describe the behaviour dynamics. In this approach, the user is assimilated to a computational neuron, and its interests are deduced by counting potential spike trains, generated by external currents. The main novelty of our approach consists in resorting to clustering task to discover natural groups, which are used in the next step to verify the neuronal response and to tune the computational model. Preliminary experimental results, based on a phantom database and obtained from a real world scenario, are shown. To discuss the obtained results, we report a comparison between the cluster memberships and the spike generation; our approach resulted to perfectly model cluster assignment and spike emission.*

[21] **De Michele P.** et al.: *A Biologically Inspired Model for Analyzing Behaviours in a Social Network Community and Cultural Heritage Scenario.*

Abstract. *In this paper, a biologically inspired mathematical model to simulate social network behaviours is presented. We propose a computational network of* Integrate & Fire *neurons to model the dynamics of spectators in a community that shares interests on cultural heritage assets. In our approach, the users are assimilated to neurons, while the friendships and the common interests are the synapses (i.e., connections) among them. These connections may be more or less strong according to how interests to a particular feature are shared. The main novelty consists in describing the propagation of*

the information on the network by simulating various applicative scenarios. Preliminary experimental results, based on a phantom database and on real world data, are shown.

4.2 Introduction

In the cultural heritage area, the needs of innovative tools and methodologies to enhance the quality of services and to develop smart applications is an increasing requirement. Cultural heritage systems contain a huge amount of interrelated data that are more complex to classify and analyse.

For example, in an art exhibition, it is of great interest to characterize, study, and measure the level of knowledge of a visitor with respect to an artwork, and also the dynamics of social interaction on a relationship network. The study of individual interactions with the *tangible culture* (e.g., monuments, works of art, and artefacts) or with the *intangible culture* (e.g., traditions, language, and knowledge) is a very interesting research field.

To understand and to analyse how artworks influence the social behaviours are very hard challenges. Semantic web approaches have been increasingly used to organize different art collections not only to infer information about an opera, but also to browse, visualize, and recommend objects across heterogeneous collections [80]. Other methods are based on statistical analysis of user datasets in order to identify common paths (i.e., *patterns*) in the available information. Here, the main difficulty is the management and the retrieval of large databases as well as issues of privacy and professional ethics [64]. Finally, models of artificial neural networks, typical of Artificial Intelligence field, are adopted. Unfortunately, these approaches seems to be, in general, too restrictive in describing complex dynamics of social behaviours and interactions in the cultural heritage framework [62].

In this Chapter, we propose a comparative analysis for classification and clustering approaches, in order to discover a reliable strategy to tune the model parameters. Specifically, we adopt two different strategies to discover data groups: the first one consists in exploiting the supervised data groupings by means of a Bayesian classifier [22], whereas the second one is based on a new approach that finds data groupings in an unsupervised way [24]. Such a strategy resorts to a *clustering* task employing the well-known K-means algorithm [54]. The main purpose of our research has the aim of underlining the advantages of the clustering-based approach with respect to the one based on the Bayesian classifier in producing data groups (i.e., *clusters*) that highlight hidden patterns and previously unknown features in the data. This fact naturally impacts on the following step, which consists in estimating the values characterizing neuron electrical properties

of the adopted network model. Such a discovered mathematical model is particularly suitable to analyse visitor behaviours in cultural assets [8, 14, 20].

Here we refer to a computational neuroscience terminology for which a cultural asset visitor is *a neuron* and its interests are *the electrical activity* which has been stimulated by appropriate currents. More specifically, the dynamics of the information flows, which are the social knowledge, are characterized by neural interactions in biological inspired neural networks. Reasoning by similarity, the users are the neurons in a network and its interests are the morphology; the common topics among users are the neuronal synapses; the social knowledge is the electrical activity in terms of quantitative and qualitative neuronal responses (spikes). This lead to produce a characterization of user dynamics and social behaviours, starting from scenarios based on phantom and real datasets.

4.3 A validation protocol

Our research starts from the data collected in a real scenario. The key point event was an art exhibition within *Maschio Angioino Castle*, in Naples (Italy), of sculptures by Francesco Jerace, promoted by DATABENC (`http://www.databenc.it`), a High Technology District for Cultural Heritage management recently founded by Regione Campania (Italy). The sculptures was located in three rooms and each of them was equipped with a sensor, able to "talk" with the users. After the event, the collected data have been organized in a structured knowledge entity, named "booklet" [13]. The booklet contents are necessary to feed the artworks fruition and they require a particular structure to ensure that the artworks start to talk and interact with the people. The Listing 4.1 shows a XML schema diagram of a simplified model of the booklet entity, characterized by the attributes of an artwork.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2   <USER ID='UI001'>
3           <STEREOTYPE_USER>2</STEREOTYPE_USER>
4           <START_SESSION></START_SESSION>
5           <END_SESSION></END_SESSION>
6    <TRANSACTION>
7     <REQUEST>
8       <HTTP_METHOD>GET</HTTP_METHOD>
9       <PATH_INFO>/opera</PATH_INFO>
10      <REQUEST_PARAMETERS>
11        <CODEARTWORK>ART0224VICTA</CODEARTWORK>
12        <DATE>13/05/2013</DATE>
13          </REQUEST_PARAMETERS>
14        <REMOTE_ADDRESS>192.168.1.6</REMOTE_ADDRESS>
15   </REQUEST>
16     <PARAMETERS_LOG>
17          <HOUR_LISTEN_START>13/05/2013 13:58:12</HOUR_LISTEN_START>
```

```
18          <HOUR_LISTEN_END>13/05/2013 14:05:42</HOUR_LISTEN_END>
19          <AUDIOS>
20      <TOT_NUMBER>3</TOT_NUMBER>
21      <AUDIO ID='AU1111'>
22              <HOUR_END>13/05/2013 14:00:42</HOUR_END>
23       <LENGTH>180</LENGTH>
24    </AUDIO>
25        </AUDIOS>
26          <IMAGES>
27      <TOT_NUMBER>11</TOT_NUMBER>
28      <IMAGE  ID='IM1122' />
29      <IMAGE  ID='IM1134' />
30      <IMAGE  ID='IM1135' />
31        </IMAGES>
32          <VIDEOS>
33      <TOT_NUMBER>2</TOT_NUMBER>
34      <VIDEO ID='VI3333'>
35              <HOUR_END>13/05/2013 14:20:12</HOUR_END>
36       <LENGTH>180</LENGTH>
37    </VIDEO>
38        </VIDEOS>
39          <TEXTS>
40      <TOT_NUMBER>4</TOT_NUMBER>
41      <TEXT ID='TX4455' />
42              <TEXT ID='TX4456' />
43          <TEXT ID='TX4457' />
44        <TEXT ID='TX4458' />
45 </TEXTS>
46      </PARAMETERS_LOG>
47   </TRANSACTION>
48 </USER>
```

LISTING 4.1: An example of the structured LOG file.

We analyse the *log file* of a phantom database that was populated with both real and random data. It represents the basic knowledge on which we test the applicability of the proposed biological inspired mathematical model. More specifically, in our experiments, we adopted the *Integrate & Fire* model (see Chapter 1.5.3) and our goal is to apply the discussed model to a case study of an artwork visitor of a cultural heritage asset in an exhibit.

Here we show the details of the investigated validation protocol to discover neuronal network model. The first approach is supervised and consists in the adoption of a naive Bayesian classifier.

We have organized the log file structure (Listing 4.1, discussed in the Section 4.3, in a Weka's `ARFF` file format (Weka, Data Mining Software in Java, `http://www.cs.waikato.ac.nz/ml/weka/`) and we have used it as an input for the Bayesian classifier. In the following, we report an example of the typical adopted `ARFF` file.

```
@RELATION ARTWORK
@ATTRIBUTE audios {p02, p04, p06, p08, p1}
@ATTRIBUTE images {p02, p04, p06, p08, p1}
@ATTRIBUTE texts {p02, p04, p06, p08, p1}
@ATTRIBUTE spike {yes, no}
@DATA
p02,p02,p02,No
p04,p02,p02,No
...
p1,p04,p02,Yes
...
```

In the proposed scheme, the values

$$p02,\ p04,\ \ldots,\ p1$$

are the percentile of a fixed `ATTRIBUTE` (audios, images or texts). For example, the value $p02$ of the feature *images* means that the user has viewed almost the 20% of overall images available for the specific artwork. A special role is played by the `ATTRIBUTE spike` that reports the interest about an opera w.r.t a suitable feature combination. More in details, the tuple $p1, p04, p02$ means that the user has listen between the 80% and 100% of the available audios, see between the 20% and 40% of the available images, read between the 0% and 20% of the available texts and, in this case, *spike* is equal to yes, i.e. user is interested to the artwork.

We recall that we are interested to find the I&F dynamic correlation with the output of a such well-known classification method. Then, in order to have a comparison metric with the results returned from the model, we choose to analyse the data trough a naive Bayesian classifier. The selected one is fairly intuitive and is based on the minimization of the following cost function:

$$CM(x_1, x_2, \cdots, x_n) =$$
$$= \arg\max_z p\big(Z = z\big) \prod_{i=1}^{N} p(X_i = x_i | Z = z)$$

where Z is a dependent class variable and $X_1 \cdots X_n$ are several feature variables. The classifier is based on the computation of individual conditional probabilities for each values of the class variable Z and for each feature $p(X_i|Z_j)$. The class, given by Bayesian classifier, is the one for which we have the largest product of the probabilities. The

Metric	**Result**
# of elements to classify	125
# of True Positive	65
# of False Positive	0
# of True Negative	52
# of False Negative	8
Precision	1
Recall	0.89

TABLE 4.1: BC classifier metrics.

Maximum Likelihood Estimation Method [71] is used to determine the individual conditional probabilities.

We use the Bayesian classifier for investigate the data and in the Table 4.1 we report some output. In this way, combining the values of the attributes `audios`, `images` and `texts`, it is possible to obtain a total of $N = 125$ different tuples, belonging to the set called U. Assuming split these tuples into two classes: let be C the class of the tuples that involve a "spike", namely the class of tuples to which the value `yes`, of the attribute `spike`, is associated; let be $U \setminus C$ the class of the tuples that do not involve a "spike", namely the class of tuples to which the value `no`, of the attribute `spike`, is associated.

At the end of the classification process, on all the N elements, for which the actual classification is known, it is possible to define the following parameters:

- **True Positive** (TP), that is the number of elements that were classified as belonging to C and actually belong to C.
- **False Positive** (FP), that is the number of elements that were classified as belonging to C but that, in reality, belong to $U \setminus C$.
- **True Negative** (TN), that is the number of elements that were classified as belonging to $U \setminus C$ and actually belong to $U \setminus C$.
- **False Positive** (FP), that is the number of elements that were classified as belonging to $U \setminus C$ but that, in reality, belong to C.

Obviously, $TP+FP+TN+FN = N$ and the number of elements that the classification process has classified as belonging to C is $TP + FP$. Then, it is possible to define the two metrics needed to evaluate the quality of the classification: **recall** and **precision**, which are

$$\frac{TP}{(TP + FN)} \qquad \frac{TP}{(TP + FP)}$$

Note that both precision and recall ranges in [0..1]. All the above mentioned parameters are reported in the Table 4.1. Moreover, we propose a more finest strategy to discover classes in the data which can be used for the next modelling step, which is the tuning of the electrical parameters for the circuit model characterizing the neuron. In fact, classification algorithms have the major limitation of labelling data according to a yet-known training set, as they are supervised approaches. In many real world datasets, data objects do not typically have assigned class membership, and this may lead to have accuracy issues in the whole classification process.

For this reason, we propose to address such an issue by introducing a *clustering*-based approach [37, 54, 58] to discover data groups. Clustering is an *unsupervised* task, since it can be applied to unclassified data (i.e., unlabelled) to obtain homogeneous object groupings. In this approach, groups are more representative with respect to single object as they summarize their common features and/or patterns; indeed, objects belonging to the same group are quite similar each other, whereas objects in different groups are quite dissimilar.

In our context, data to be clustered are *tuples* representing visitor's behaviours related to an artwork. Note that now "spike" has a more informative role in the dataset, as it is not seen as a class but as a further information about visitor's behaviour. In our experiments, we assume the following criteria for spike generation. A visitor enjoyed an artwork if he benefits from the whole content of at least one of the available services, or if he exploits more than the 66% of the total contents.

This new clustering-based approach allows us to produce a more general dataset, in which we do not need to assign object classes, and also attributes can take values in a continuous range, instead of in a discrete one. Therefore, the clustering phase produces groups according to visitor's preferences, which are not necessary driven by spike generation.

On these hypothesis, we have rearranged the log file structure (Listing 4.1), in a suitable way for clustering process, as follow:

```
@RELATION ARTWORK
@ATTRIBUTE audios NUMERIC [0..1]
@ATTRIBUTE images NUMERIC [0..1]
@ATTRIBUTE texts NUMERIC [0..1]
@ATTRIBUTE spike {0,1}
@DATA
0.1,0.4,1.0,1
0.3,0.6,0.4,0
...
0.5,1.0,0.7,1
...
```

In the proposed scheme, data values represent the amount of information that the visitor has exploited for an artwork for each attribute of the dataset, and the last attribute describes the spike generation according to the algorithm previously described. In this way, combining the values of the attributes `audios`, `images` and `texts`, it is possible to obtain a total of $N = 1,331$ different data objects (i.e., tuples) — for simplicity, we take into account just real values rounded at the first decimal value.

As regards the clustering task, we can employ any algorithm to discover groups. However, in this paper, we resorted to the well-known K-means clustering algorithm [54]. K-means requires only one parameter, that is the number K of clusters (i.e., groups) to be discovered. Algorithm 10 shows the outline of the K-means clustering algorithm.

Algorithm 10 K-means

Require: a dataset objects $\mathcal{D} = \{o_1, \ldots, o_N\}$; the number of output clusters K
Ensure: a set of clusters $\mathcal{C} = \{C_1, \ldots, C_K\}$

1: **for** $i = 1$ to K **do**
2: $c_i \leftarrow$randomInitialize($\mathcal{D}$)
3: **end for**
4: **repeat**
5: **for all** $C_i \in \mathcal{C}$ **do**
6: $C_i \leftarrow \emptyset$
7: **end for**
8: **for all** $o_u \in \mathcal{D}$ **do**
9: $j \leftarrow argmin_{i \in [1..K]} dist(o_u, c_i)$
10: $C_j \leftarrow C_j \cup \{o_u\}$
11: **end for**
12: **for all** $C_i \in \mathcal{C}$ **do**
13: $c_i \leftarrow$ updateCentroid(C_i)
14: **end for**
15: **until** centroids do not change or a certain termination criterion is reached

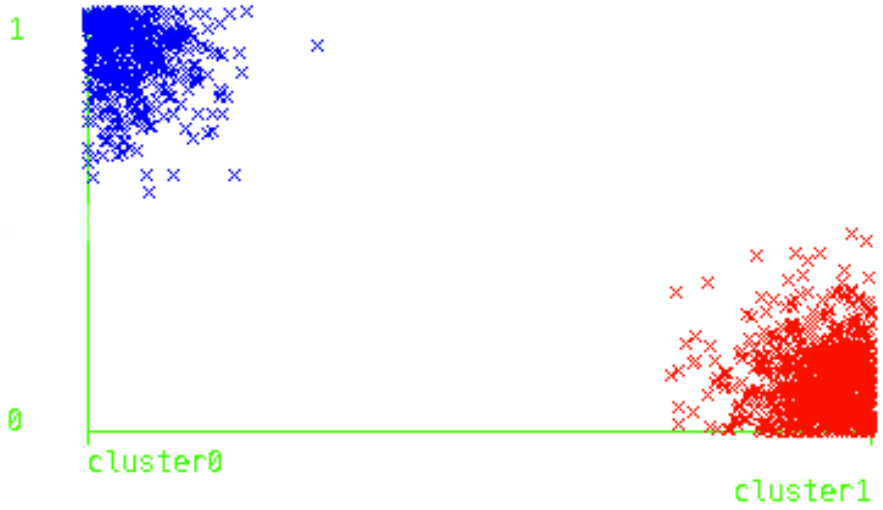

FIGURE 4.1: Clustering results for K-means ($K = 2$)

In our experiments, we first started with $K = 2$, which is the natural starting choice to model a classification-like approach (i.e, "spike" or "no-spike"). Nevertheless, we can also perform further experiments by setting higher values for K to capture finest similarities and/or hidden patterns in the data. Figure 4.1 shows the output of the clustering phase with $K = 2$. Note that we do not take into account the "spike" attribute in the clustering process, as it could clearly bias the entire process. However, we exploited it at the end of the clustering phase to assess the result accuracy. We resorted to Weka "simpleKMeans" implementation, and the plot is also obtained employing Weka clustering visualization facilities.

The plot represents tuples in terms of cluster membership (x-axis) and spike emission (y-axis). It is easy to note that all the data in *cluster0* refer to tuples that produce spikes (i.e., with value 1), whereas all the ones in *cluster1* identify tuples that do not emit spike (i.e., with value 0). Therefore, evaluating clustering results in terms of well-separation of the data with respect to the spike emission issue, we achieved a high-quality clustering as all the data have been correctly separated.

4.4 The single neuron model

By comparing the two proposed approaches, it is easy to note that the one based on clustering furnishes more significant groups, which are more homogeneous in terms of spike emission. In fact, summary results for the Bayesian-based model are shown in Table 4.1, which highlights a certain number of False Negative data (i.e., $FN = 8$). Consequently, this fact negatively affects Recall value, that is equal to 0.89. On the contrary, the clustering-based approach does not assign any false membership in the

Tuples	**M.C.F. (%)**	**Cluster**	**# spikes**
$0.2, 0.2, 0.2$	20%	cluster1	0
$0.2, 0.2, 0.4$	27%	cluster1	0
$0.4, 0.2, 0.2$	27%	cluster1	0
$0.6, 0.6, 0.7$	63%	cluster1	0
$0.6, 0.8, 0.8$	73%	cluster0	4
$0.7, 0.9, 0.5$	70%	cluster0	4
$0.8, 0.9, 0.3$	67%	cluster0	2
$0.8, 0.9, 0.6$	76%	cluster0	5
$1.0, 0.2, 0.1$	$43\%^{(*)}$	cluster0	5
$1.0, 0.8, 0.9$	$90\%^{(*)}$	cluster0	10
$1.0, 1.0, 0.6$	$86\%^{(*)}$	cluster0	13
$1.0, 1.0, 1.0$	$100\%^{(*)}$	cluster0	16

TABLE 4.2: Spike response for clustering and I&F model with $(R, C) = (0.51kOhm, 30\mu F)$.

data. This naturally impacts on our purpose of identifying user behaviours. For these motivations, we adopted the clustering task to guide model parameter identification.

Then, starting from the clustering output, we have integrated the I&F computational model in order to find some correlations with the clustering results. In particular, the couple (R, C) represents the visitor sensitivity to the artwork. We have exploited the clustering results in order to tune the values of the resistance R and conductance C of the circuit that represents the model. In a first experiment, a good choice for the couple (R, C) is

$$(R, C) = (0.51kOhm, 30\mu F)$$

The current is a linear combination of the values of the attributes in the dataset. The Figure 4.2 gives the dynamic response of the neuron.

In the first case (top of the Figure 4.2) the current $I(t)$ is not sufficient to trigger a potential difference which gives a spike. In the second one (bottom of the Figure 4.2) the neuron that has received stimuli is able to produce an interesting dynamic.

In these experiments, we show how the computational model and the clustering give information about the interest of a visitor about an artwork. In the Table 4.2, experimental results for the clustering and our model are reported. M.C.F. represents the Media Content Fruition with respect to the overall media contents. With the symbol (*) we have labelled the tuple combinations that contain the information about the fully fruition of at least one media content. Note that the last column of the table indicates the degree of the visitor interest for an artwork. Thus, in this respect, such an information is obtained by the proposed I&F neuron model to achieve a fine-grained indication for spikes.

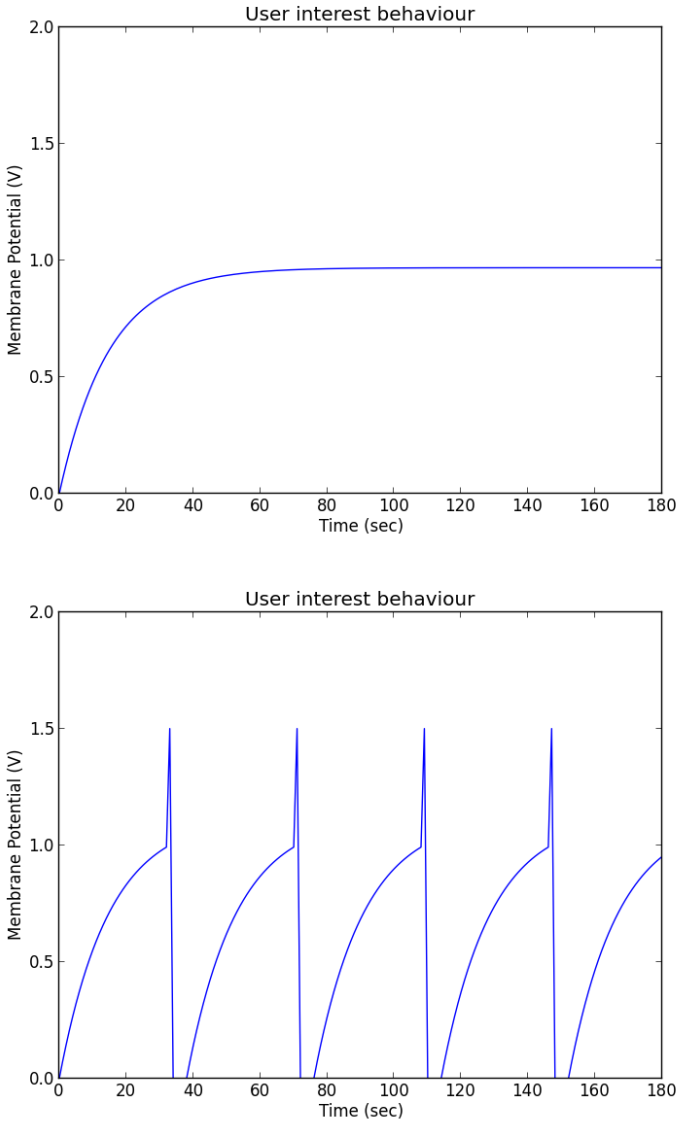

FIGURE 4.2: **Top**. With a current $I(t) = 0.6 + 0.6 + 0.7$, the neuron has no spikes. **Bottom.** With a current $I(t) = 0.6 + 0.8 + 0.8$ the neuron has 4 spikes.

In Figure 4.3, we have fixed

$$I(t) = 0.8 + 0.9 + 0.3$$

as a stimulus and we have compared two users U_1 with $(R, C) = (0.51, 30)$ and U_2 with $(R, C) = (0.6, 28)$.

We can observe the different number of spikes between U_1 and U_2 respect to the same artwork. If the spike are related to the the interests that a cultural asset has aroused in a viewer, the I&F is able to emerge this features. The choice of the pair (R, C) suitable for a established user is the real challenge of the model. More in general, it may be multiple scenarios to apply these dynamics. An example is the case of a cultural asset exhibition in which the target is how to place artworks. A possible choice is to select the

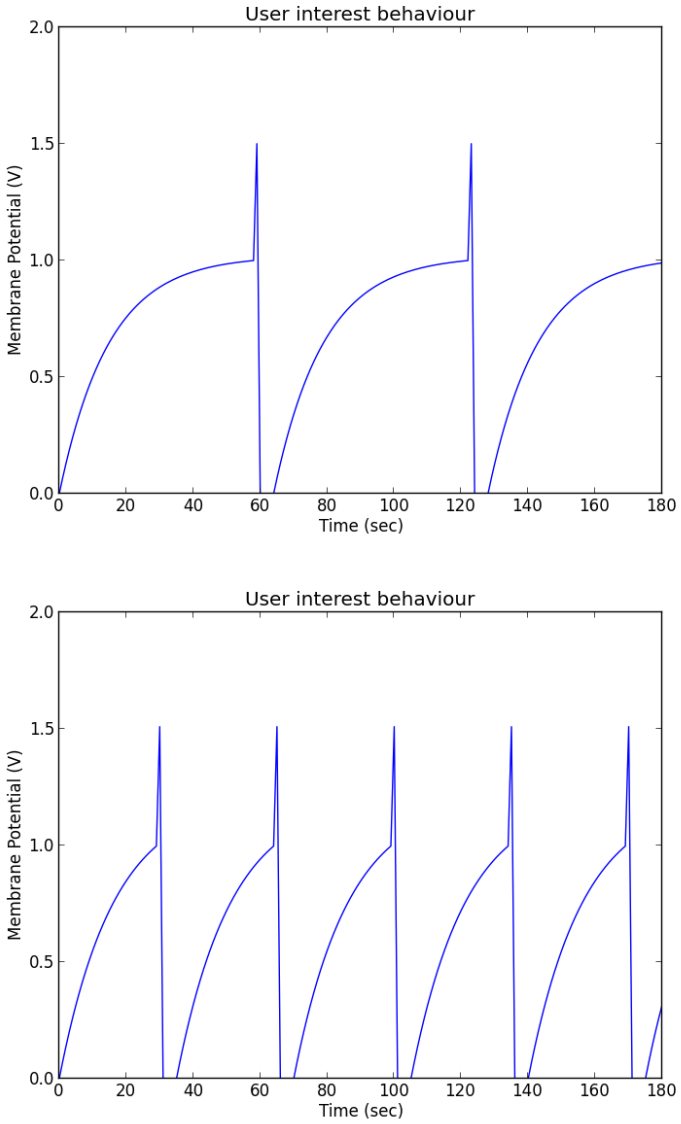

FIGURE 4.3: **Top**. With the couple $(R, C) = (0.51, 30)$ the neuron has 2 spikes. **Bottom.** With the couple $(R, C) = (0.6, 28)$ the neuron has 5 spikes.

operas that have attracted the visitors with common interests, i.e., users with similar (R, C). In the context-aware profiling instead the aim is how to change (R, C) in such a way to predict the user behaviours in terms of spikes that represent its cultural assets.

4.5 The network model

Preliminary, we apply the discussed model to a case study of an artwork visitor of a cultural heritage asset in an exhibit. Here the neural network represents a social network and neurons represent the users. Neurons with synapses are intended such as users which have a *friendship* or which belong to the same *circle* (i.e., *fan page*). We

introduce a neural network model, which uses leaky I&F neurons. In this case, with respect to the single neuron model, there are more components to take into account. In a neural network neurons are connected among them by means of synapses. The neuron n_i is linked to n_j by means of a synapse that is a positive weight $w_{i,j}$. In other terms, we model the network morphology by means a matrix

$$W = \begin{cases} w_{i,j} > 0 & if \;\; n_i \;\; is \;\; linked \;\; to \;\; n_j \\ 0 & otherwise. \end{cases}$$

Note that, in our case, we assume a symmetric structure for the matrix W, i.e. $w_{i,j} = w_{j,i}$.

In a network, the neuron is stimulated by pre-synaptic currents arriving at its synapses. Each pre-synaptic current gives a stereotyped contribution, described by a function $\alpha(t)$, to the post-synaptic current. The contributions of different pre-synaptic spikes are linearly combined to have the total post-synaptic current. In particular, the total post-synaptic current to the n_i neuron is modelled as:

$$I_i(t) = \sum_j w_{i,j} \sum_k \alpha(t - t_j^{(k)})$$

where $t_j^{(k)}$ is the time of the $k-$th spike of the $j-$th pre-synaptic neuron.
The weights $w_{i,j}$ are a measure of the *liking degree* of the users with respect to an assigned feature, for example the interest on the art history, on an artistic period or on an artwork. In the Figure 4.4 we show the matrix W that represents the connections among 348 *friends* (i.e., users) on Facebook. The dataset coming from the Stanford Network Analysis Platform (`https://snap.stanford.edu`) and we highlight the connection among the user U_0 and other 347 friends (neurons of the network),

In our previous works [22, 24] we deal with a single neuron model where the (R_m, C_m) couple represents the sensitivity of the user with respect to an artwork. In this context, the (R_m, C_m) couple gives information about the *sociality level* of a user. The measure of the sociality level is expressed by the product

$$\tau = R_m * C_m$$

and it is the tendency of the user to interact on a social network, sharing, liking or commenting a social content. For our aims it holds that

- a small value of τ corresponds to an high sociality level;
- a large value of $w_{i,j}$ gives that liking degree between i and j is high.

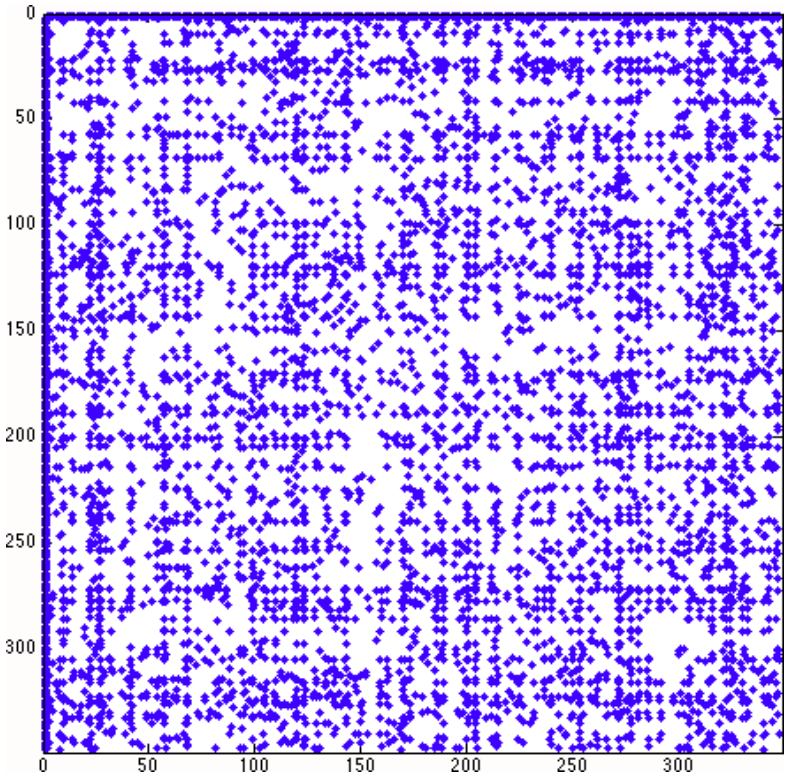

FIGURE 4.4: Connection matrix W between 348 neurons.

By using the draw framework, the neuron spikes give information about the social activity, i.e. about the sharing, the liking or commenting a social content. In our experiments only one neuron is subject to an external stimulus I_{ext} and we are interested to observe how his neural activity influences other neurons. By the social network point of view, this means that we observe how the social activity of a single user influences other users.

4.6 User behaviours on a social network

In the following experiments, an artwork spectator U_1 with $(R_{m_1}, C_{m_1}) = (0.51kOhm, 10\mu F)$ is stimulated with an external current $I_{ext} = I(t) = 1 + 1 + 1$ that corresponds to have beneficed from 100% of text, images and video of an artwork.

The Figure 4.5 shows the electrical activity in terms of spikes of this art spectator. We can affirm that the spectator is very interest to the opera and he emits 26 spikes. In the following subsections we assume that a spectator has shared this experience in a social network and we refer to users in a social network as neurons in a neural network. We are interested to analyse the network response in terms of social activity (i.e., electrical stimulations) induced on the net.

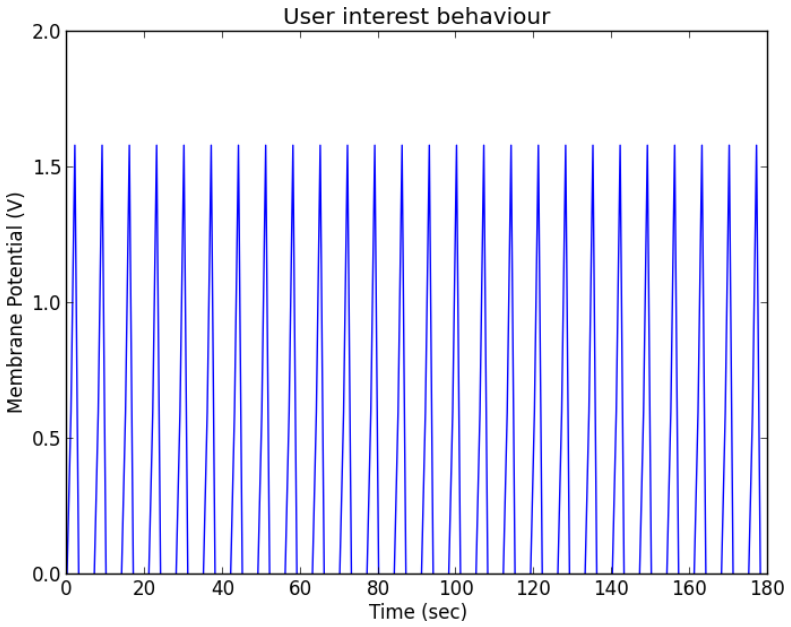

FIGURE 4.5: With the couple $(R_{m_1}, C_{m_1}) = (0.51kOhm, 10\mu F)$ the user (i.e., neuron) U_1 has 26 spikes.

User	$\tau_{\mathbf{i}}$ $(\mathbf{w_{1,i} = 1.3})$	$\tau_{\mathbf{i}}$ $(\mathbf{w_{1,i} = 0.5})$
U_1	5.10	5.10
U_2	**1.68**	**2.36**
U_3	7.94	**1.04**
U_4	**5.49**	**1.67**
U_5	8.24	7.55
U_6	22.85	21.15

TABLE 4.3: Values τ_i, where i identifies the user U_i.

4.6.1 Simple interaction

We consider 6 users (i.e., neurons), where user U_1 have a friendship (i.e., is connected) with each other user. The network connectivity is shown in Figure 4.6. A key role is played by two parameters: τ_i and $w_{i,j}$. In a first experiment, the user U_1 has $(R_{m_1}, C_{m_1}) = (0.51kOhm, 10\mu F)$, the connection weights are fixed as $w_{1,i} = 1.3$ (with $i = 2, \ldots, 6$) and the values of τ_i (with $i = 1, \ldots, 6$) differ. More in detail, the sociality level of the user U_1 is $\tau_1 = 0.51 \times 10 = 5.1$ and for remaining users U_i (with $i = 2, \ldots, 6$) we have

$$\tau_i = i \times x_i \times \tau_1$$

where x_i is a random value between 0 and 1. From the simulation, we can observe that user U_1 mostly influences all the other users with a small value of τ_i (i.e., U_2 with $\tau_2 = 1.68$ and U_4 with $\tau_4 = 5.49$) as reported in bold in the second column of the Table 4.3 and in Figure 4.7.

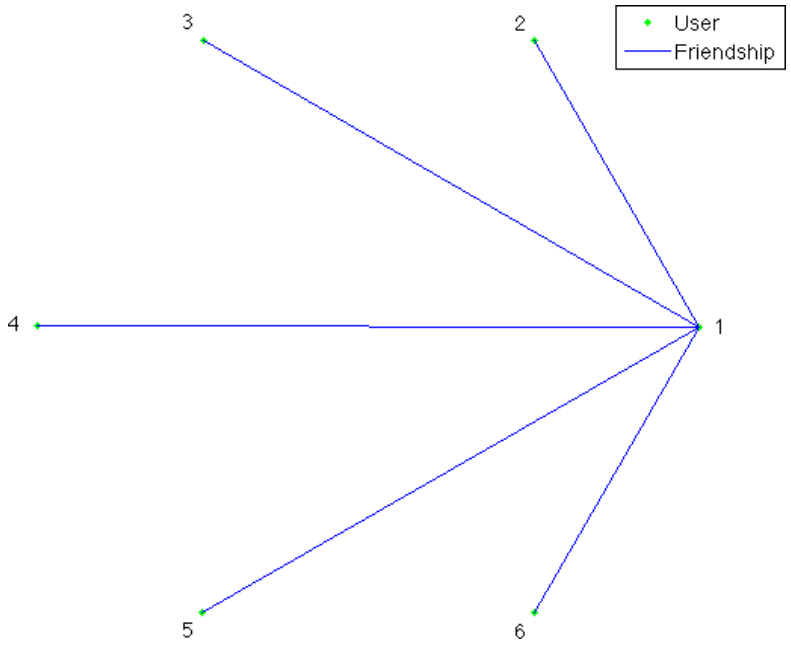

FIGURE 4.6: Social network connectivity in the case of 6 users.

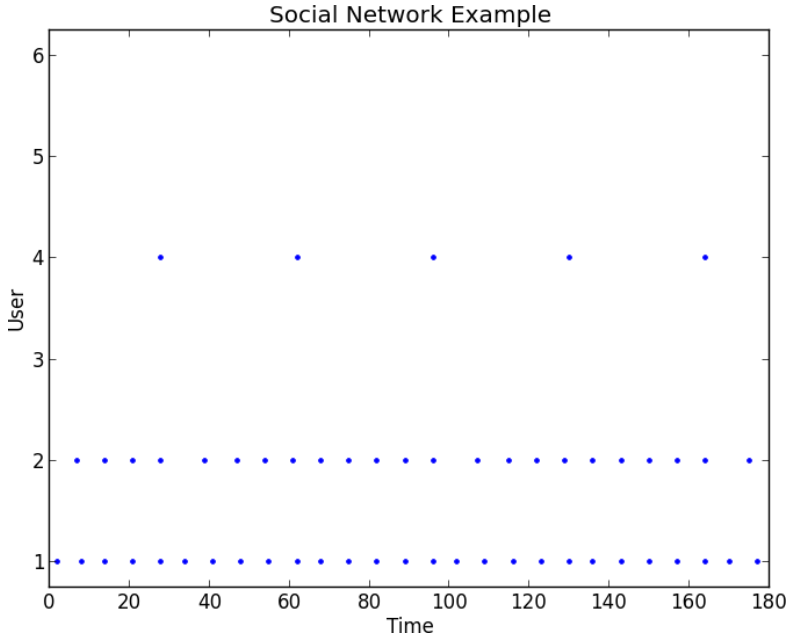

FIGURE 4.7: Social network activity in the case of 6 users, with fixed weights $w_{1,j} = 1.3$.

The behaviour of the network changes decreasing the value of the weights. In fact, if we choose $w_{1,i} = 0.5$ (with $i = 2, \ldots, 6$), the user U_1 does not influence the others, even with small values of τ_i (i.e., $\tau_2 = 2.36$, $\tau_3 = 1.04$ and $\tau_4 = 1.67$), as reported in bold in the third column of the Table 4.3 and in Figure 4.8

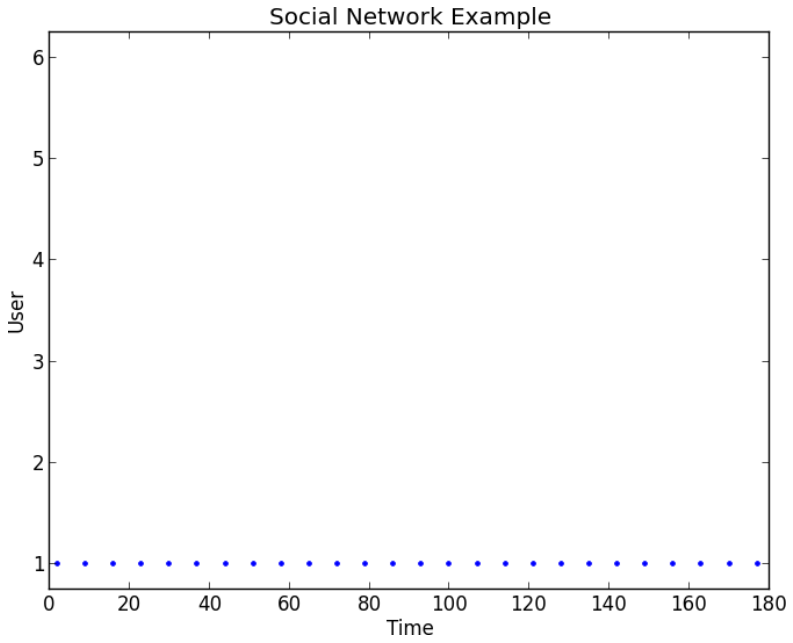

FIGURE 4.8: Social network activity in the case of 6 users, with fixed weights $w_{1,j} = 0.5$.

Synapse	$\mathbf{w_{1,i}}$ **(**$\tau_1 = 5.1$, $\tau_i = 3$**)**	$\mathbf{w_{1,i}}$ **(**$\tau_1 = 5.1$, $\tau_i = 20$**)**
$1, 2$	0.75	1.02
$1, 3$	**1.58**	1.03
$1, 4$	0.44	**1.67**
$1, 5$	**1.65**	**1.64**
$1, 6$	1.07	0.35

TABLE 4.4: Values of the synapses $w_{1,i}$, where 1 represents the user U_1 and i an user U_i.

In a second simulation, we set the sociality levels at $\tau_1 = 5.1$, $\tau_i = 3$ (with $(R_{m_i}, C_{m_i}) = (0.5, 6)$), for $i = 2, \ldots, 6$, and we change the weights as

$$w_{1,i} = x_i \times 2$$

where x_i is a random value between 0 and 1. The network topology is the same of the one shown in Figure 4.6. We can observe that the user U_1 mostly influences users with a large $w_{1,j}$ (i.e., $w_{1,3} = 1.58$ and $w_{1,5} = 1.65$) as shown in bold in the second column of Table 4.4 and Figure 4.8.

The behaviour of the network is different by increasing the value of $\tau_i = 20$ (with $i = 2, \ldots, 6$ and $(R_{m_i}, C_{m_i}) = (0.5, 40)$). In fact, in this case, we can observe from the third column of the Table 4.4 that user U_1 minimally affects the other users, even with large values of $w_{1,i}$ ($w_{1,4} = 1.67$ and $w_{1,5} = 1.64$). The social network activity is reported in the Figure 4.10.

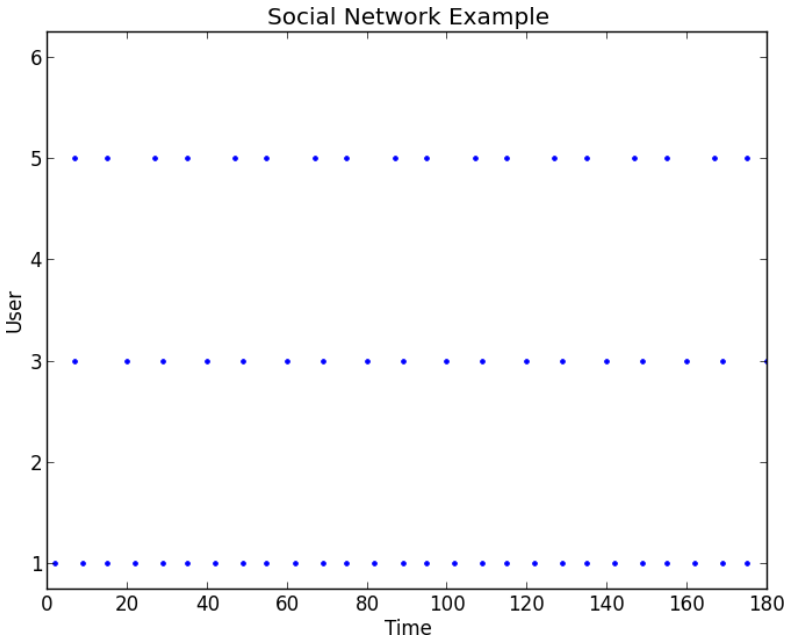

FIGURE 4.9: Social network activity in the case of 6 users, with $\tau_1 = 5.1$, $\tau_i = 3$.

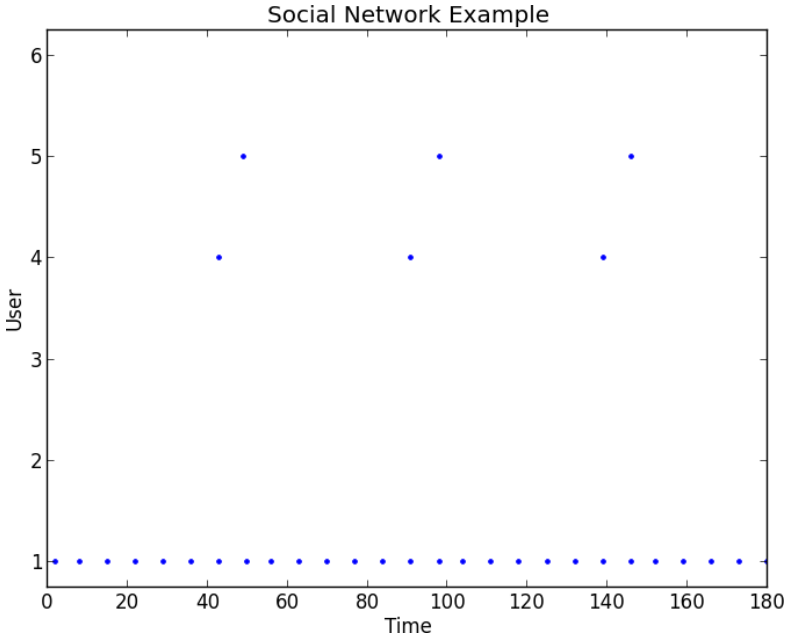

FIGURE 4.10: Social network activity in the case of 6 users, with $\tau_1 = 5.1$, $\tau_i = 20$.

4.6.2 Viral interaction

In a more complex scenario, we consider 31 users and we introduce $M = 5$ as the cardinality of the friendships. Then, users U_i, with $i = 1, \ldots, M+1$, are connected with M users U_j, where

$$j = M * (i-1) + 2, \ldots, (M+1) * i + 1 - i$$

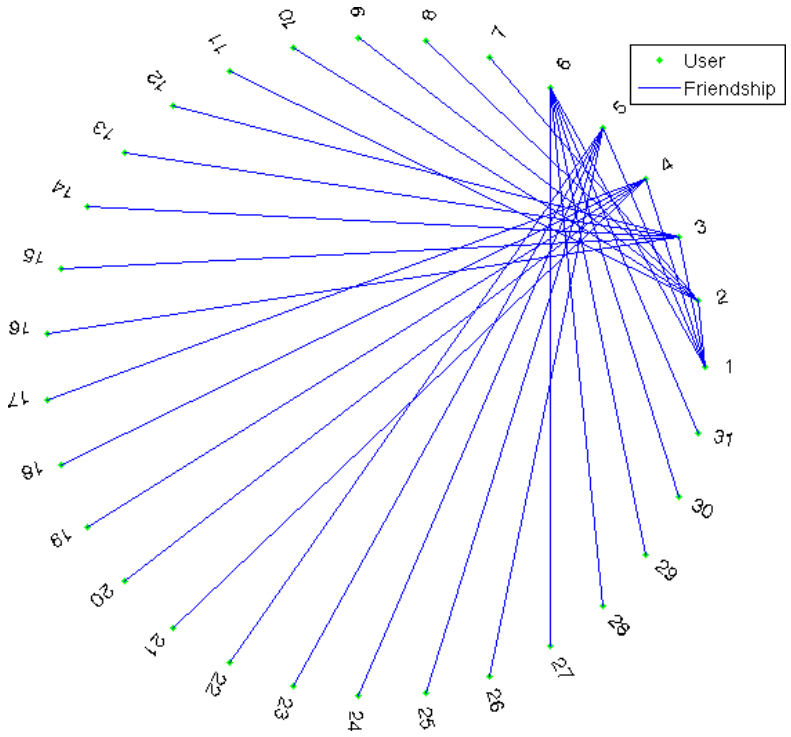

FIGURE 4.11: Social network connectivity in the case of 31 users.

as shown in Figures 4.11 and 4.12. This scenario is typical of a *viral* context, in which a social content can reach users even though they do not have a 1st level degree connection (i.e., friendship). The virality of our case study is depicted in Figure 4.12.

We fix the couple $(R_{m_1}, C_{m_1}) = (0.51, 10)$ for the user U_1 ($\tau_1 = 5.1$), the weights $w_{i,j} = 1.3$ (with $i = 1, \ldots, M+1$ and $j = M*(i-1)+2, \ldots, (M+1)*i+1-i$) and the sociality level of U_i as

$$\tau_i = i \times x_i \times \tau_1$$

where x_i is a random value between 0 and 1. Comparing the second column of the Table 4.5 with the Figure 4.13, we observe that the user U_1 mostly influences users with a small value of τ_i (i.e., U_2 with $\tau_2 = 0.40$, U_3 with $\tau_3 = 1.78$ and U_4 with $\tau_4 = 3.47$). In particular, the user U_2 influences user U_9 (with $\tau_9 = 3.32$), the user U_3 influences user U_{14} (with $\tau_{14} = 1.59$), and the user U_4 influences users U_{20} (with $\tau_{20} = 7.65$) and U_{25} (with $\tau_{25} = 2.32$).

The behaviour of the network changes by decreasing the value of the weights. If we set $w_{i,j} = 0.5$ (with $i = 1, \ldots, M+1$ and $j = M*(i-1)+2, \ldots, (M+1)*i+1-i$), observing the third column of the Table 4.5 and the Figure 4.14, the user U_1 maximally influences just the user U_2 (with a very small $\tau_2 = 0.36$), while the influence on the other users U_i (with $i = 3, \ldots, M+1$) is very poor. Note that for the users U_3 and U_4 we have also small values $\tau_3 = 1.78$ and $\tau_4 = 3.47$. Accordingly, for users U_i (with

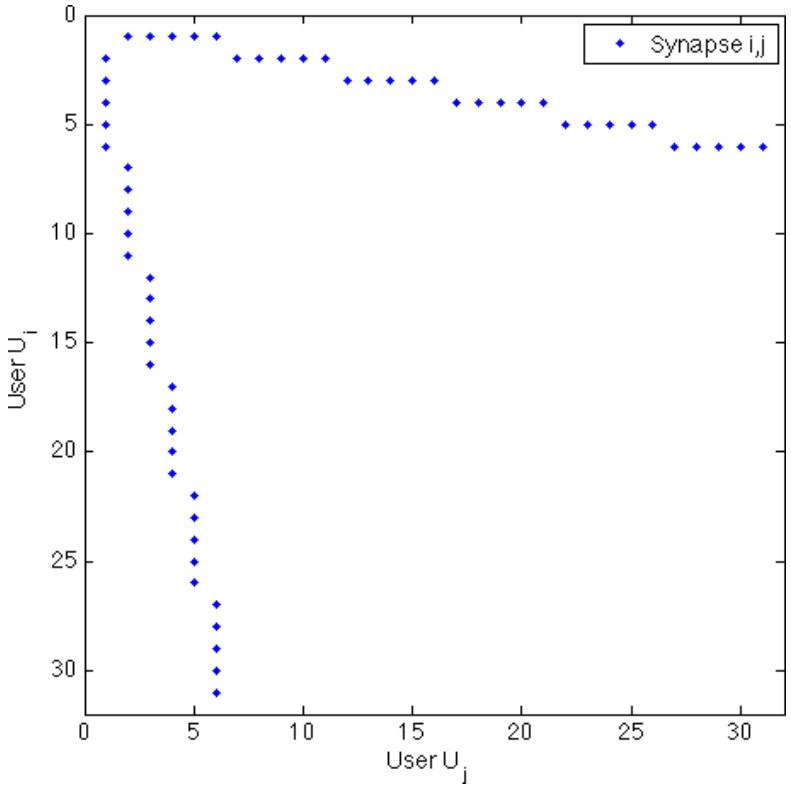

FIGURE 4.12: Different view of the social network connectivity in the case of 31 users.

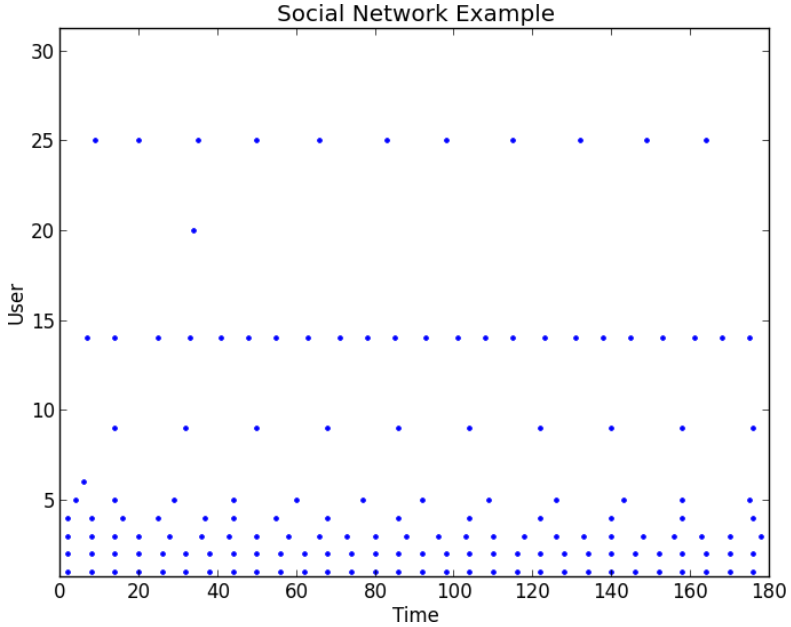

FIGURE 4.13: Social network activity in the case of 31 users, with fixed weights $w_{i,j} = 1.3$.

$i = 2, \ldots, M + 1$) the probability of influencing the other users decreases: in fact, the users U_i (with $i = M + 2, \ldots, 31$) does not have social activity.

User	$\tau_{\mathbf{i}}$ ($\mathbf{w_{i,j}} = \mathbf{1.3}$)	$\tau_{\mathbf{i}}$ ($\mathbf{w_{i,j}} = \mathbf{0.5}$)
U_1	5.10	5.10
U_2	**0.40**	**0.36**
U_3	**1.78**	6.28
U_4	**3.47**	4.07
U_5	13.56	3.18
U_6	24.79	3.99
U_7	19.76	10.57
U_8	27.26	34.31
U_9	**3.32**	23.66
U_{10}	8.39	3.72
U_{11}	17.06	17.71
U_{12}	14.13	44.86
U_{13}	44.68	39.35
U_{14}	**1.59**	53.58
U_{15}	61.96	29.28
U_{16}	48.56	10.12
U_{17}	65.15	27.47
U_{18}	68.95	49.92
U_{19}	70.66	2.18
U_{20}	**7.65**	43.35
U_{21}	11.27	82.85
U_{22}	49.08	28.21
U_{23}	53.17	82.64
U_{24}	9.72	113.94
U_{25}	**2.32**	78.98
U_{26}	108.63	0.89
U_{27}	53.21	35.29
U_{28}	106.34	75.70
U_{29}	22.75	33.48
U_{30}	23.80	33.04
U_{31}	17.74	103.30

TABLE 4.5: Values τ_i, where i identifies the user U_i.

Here, we modify the network connectivity as shown in Figures 4.15 and 4.16: now the user U_5 is connected with users U_j, where

$$j = M * 2 + 2, \ldots, (M + 1) * 5 + 1 - 5.$$

Moreover, we fix $w_{i,j} = 2.3$ (with $i = 5$, $j = 1, M * (i - 1) + 2, \ldots, (M + 1) * i + 1 - i$), $w_{i,j} = 0.2$ (with $1 \leq i < 5$, $5 < i \leq M + 1$ and $j = M * (i - 1) + 2, \ldots, (M + 1) * i + 1 - i$) and vary τ_i (with $i = 1, \ldots, 31$).

Then, we have a strong liking degree with respect to an assigned feature between the users U_1 and U_5. As in Figure 4.17, the user U_5 have an intense social activity (24

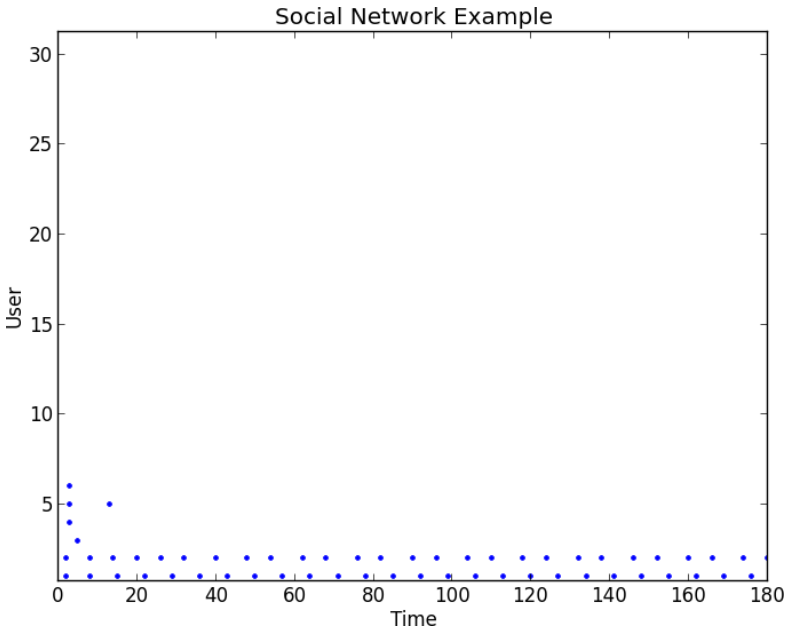

FIGURE 4.14: Social network activity in the case of 31 users, with fixed weights $w_{i,j} = 0.5$.

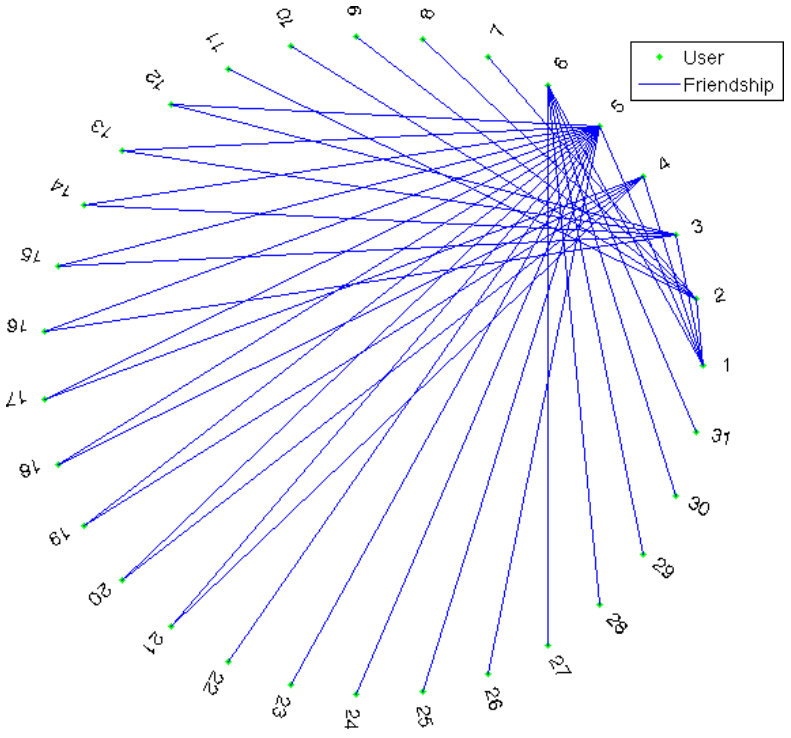

FIGURE 4.15: Social network connectivity in the case of 31 users.

spikes), while other users connected with U_1 do not have social activity. On the other hand, user U_5 mostly influences users connected to him.

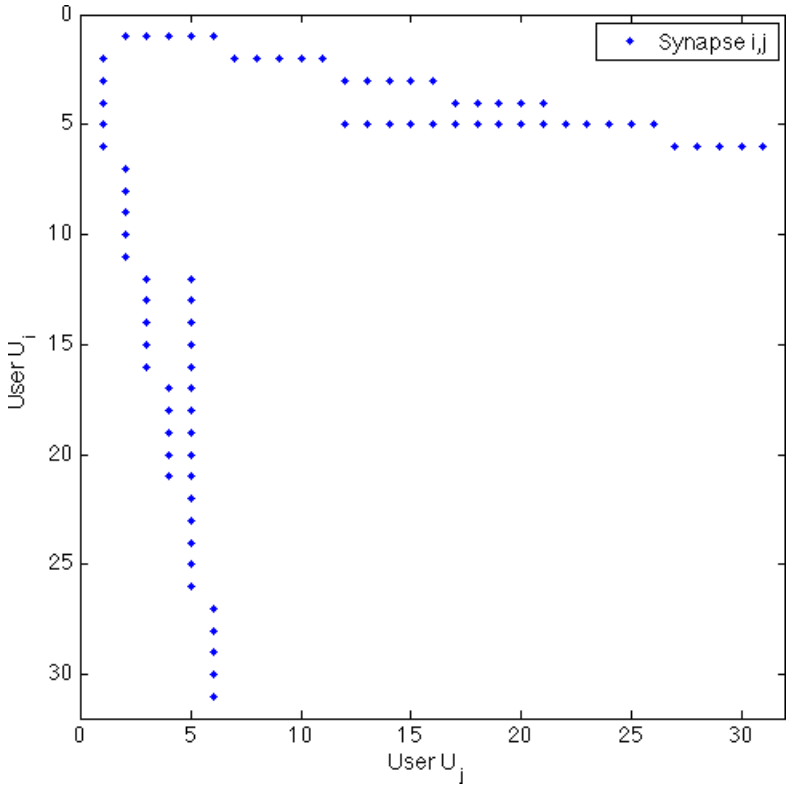

FIGURE 4.16: Different view of the social network connectivity in the case of 31 users.

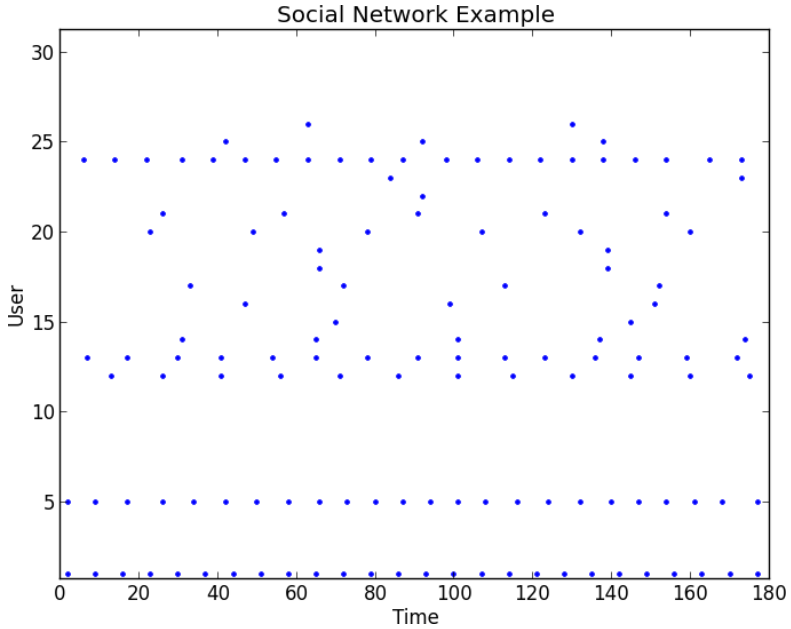

FIGURE 4.17: Social network activity in the case of 31 users, with fixed weights $w5,j = 2.3$ (with $j = 1, M*(i-1)+2, \ldots, (M+1)*i+1-i$) and the others $w_{i,j} = 0.2$.

4.6.3 Real interaction

We simulate the real case of 348 users of Facebook social network, as reported in Section 4.5 and in Figure 4.4. More in detail, the user U_1 has a friendship with all other 347

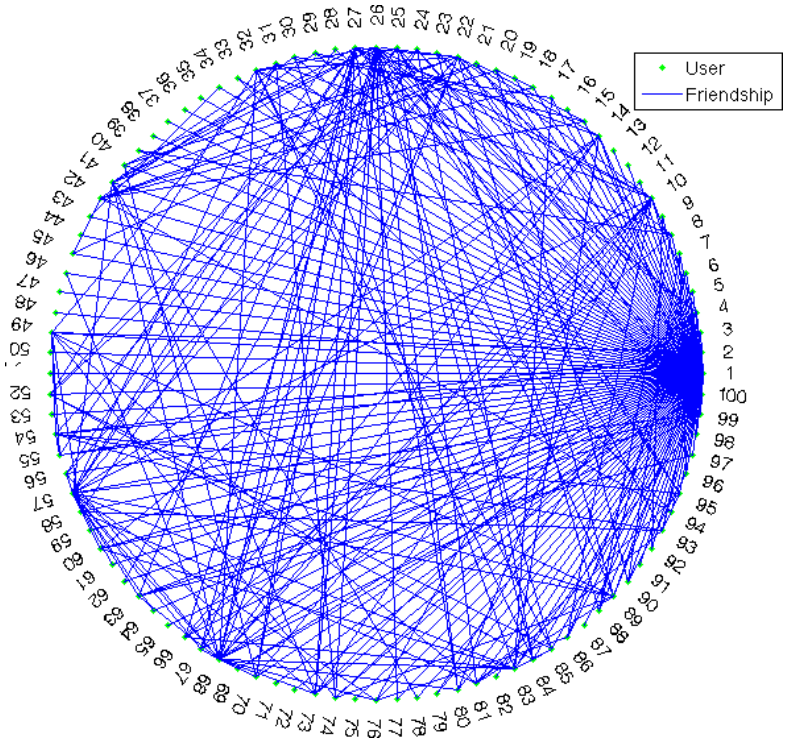

FIGURE 4.18: Social network connectivity in the case of 348 users.

users, which also can have friendship among them. If there exists a connection between two users U_i and U_j, the weight is set as $w_{i,j} = 0.2$. Moreover, users that share the interest for a particular feature with user U_1 belong to a particular circle. In this case, synapses between two users U_i and U_j belonging to the circle are incremented to the value $w_{i,j} = 0.4$. We are interested to reproduce the interactions among user U_1 with all other users and how these are influenced. In Figure 4.18, for representation simplicity, we highlight the network connectivity among the first 100 users. As we can observe, in the proximity of the user U_1 there is a high concentration of connections. This is due to the fact that user U_1 is connected with all other users.

As shown in Figure 4.19, we have 10 of the 348 users belonging to the circle: in particular, there are the users U_1, U_{100}, U_{117}, U_{141}, U_{145}, U_{148}, U_{151}, U_{156}, U_{271} and U_{328}.

In Figure 4.20 the network activity is shown. As we can observe, user U_{117} has a high number of "spikes". This is due to two facts: firstly, this user belong to the circle together U_1, then the liking degree (i.e. weight) between these users is $w_{1,117} = 0.4$; secondly, user U_{117} has a very small value of $\tau_{117} = 0.44$, then the sociality level of this user is very high.

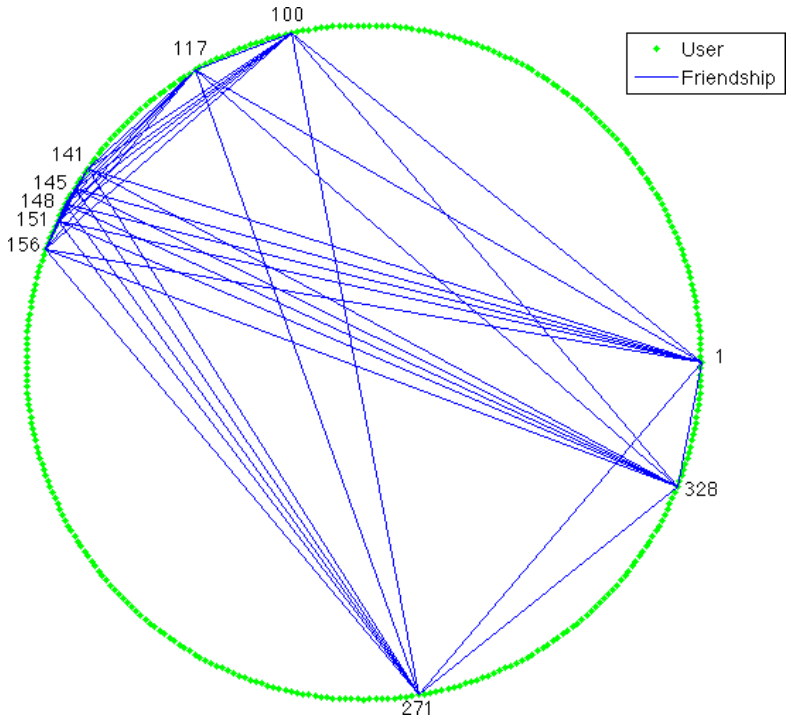

FIGURE 4.19: Social network connectivity for 10 of the 348 users.

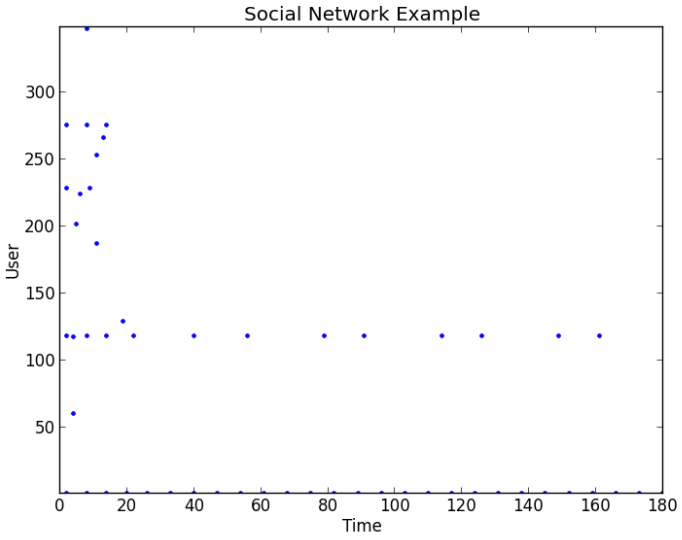

FIGURE 4.20: Social network activity for 10 of the 348 users.

4.7 Implementation details

Here, we provide some technical details regarding the implementation of the two models. From Section 1.5.3, rearranging the Eq. (1.8) we obtain

$$\frac{dV}{dt} = -\frac{V}{RC} + \frac{I(t)}{C} \tag{4.1}$$

Then, replacing $dV = V' - V$ into the Eq. (4.1) we have

$$V' = V + \left(-\frac{V}{RC} + \frac{I(t)}{C}\right)dt \tag{4.2}$$

Finally, from the the Eq. (4.2), we obtain

$$V' = V + \left(\frac{-V + I(t)R}{RC}\right)dt \tag{4.3}$$

The Eq. (4.3) is the core of our models. In Algorithm 11 we reported the code of the single neuron model, where `V[i]` is V', `V[t-1]` is V and `I_t[t]` is $I(t)$. Moreover, `time` indicates the duration of the simulation, `t` represents the time t, `threshold_rest` is the initial refractory time, `R` is the resistance R, `tau` is the time constant $\tau = R \times C$, `dt` indicates the simulation time step in ms, `V_threshold` is the spike threshold, `V_spike` is the spike delta and `tau_ref` represents the refractory period.

Algorithm 11 A light view of the core of the single neuron model.

```
...
for t in enumerate(time):
  if t > threshold_rest:
   V[t] = V[t-1] + (-V[t-1] + I_t[t]*R) / tau * dt
  if V[i] >= V_threshold:
   V[t] += V_spike
   threshold_rest = t + tau_ref
   ...
...
```

The models are implemented in Python (v. 2.7.8), and we resorted to a set of Python libraries, which are `math` for the mathematical functions, `numpy` for the list management (array), and `pylab` for the plotting.

4.8 Conclusions

We described a framework that reflects the computational methodology adopted to infer information about visitors in a cultural heritage context. The challenge here is to map, in a realistic way, the biological morphology of a neuron in this application scenario. We deal with a model where the (R, C) couple represents the sensitivity of the user respect to an artwork.

We compared two different strategies for tuning model parameters, in order to find an accurate approach that is able to provide the best setting for the neuronal model. In this

respect, we shown experimental results for classic Bayesian classifier and new clustering methodology to obtain starting groups from which these electrical parameters can be tuned. From our experiments, it has been highlighted that clustering task is able to produce a more accurate setting.

Moreover, we described a framework and the computational methodology adopted to infer information about users of a social network in a cultural heritage context. We have built a computational neural network able to reproduce the interactions in a cultural heritage community. The challenge was to map, in a realistic way, the biological morphology of the network and to deeply investigate the related parameters, that are the sociality level τ and the friendship connection matrix W. The main novelty of our work was to analyse how the information shared by a user can have influences on other actors in a circle of a social network with cultural heritage topics. An interesting observation and challenge for future works is to adapt, in a smart way, this computational framework to many different application topics, such as the context-aware profiling, feedback based and/or recommendation systems.

Bibliography

[1] P. Andersen, R. Morris, D. Amaral, T. Bliss, and J. O'Keede. *The Hippocampus Book*. Oxford: University Press, 2007.

[2] M. Barad, R. Bourtchouladze, D.G. Winder, H. Golan, and E. Kandel. A type iv-specific phosphodiesterase inhibitor, facilitates the establishment of long-lasting long-term potentiation and improves memory. *Proc. Natl. Acad. Sci. U. S. A.*, 95: 15020–15025, 1998.

[3] A. Barco and H. Marie. Genetic approaches to investigate the role of creb in neuronal plasticity and memory. *Mol. Neurobiol.*, 44(3):330–349, 2011.

[4] A. Barco, J.M. Alarcon, and E.R. Kandel. Expression of constitutively active creb protein facilitates the late phase of long-term potentiation by enhancing synaptic capture. *Cell*, 108:689–703, 2002.

[5] G. Bi and M. Poo. Synaptic modification by correlated: Hebb's postulate revisited. *Annu. Rev. Neurosci.*, 24:139–166, 2001.

[6] D. Bianchi, A. Marasco, A. Limongiello, C. Marchetti, H. Marie, B. Tirozzi, and M. Migliore. On the mechanisms underlying the depolarization block in the spiking dynamics of ca1 pyramidal neurons. *J. Comput. Neurosc.*, 33(2):207–225, 2012. doi: 10.1007/s10827-012-0383-y.

[7] D. Bianchi, A. Marasco, A. Limongiello, C. Marchetti, H. Marie, B. Tirozzi, and M. Migliore. On the mechanisms underlying the depolarization block in the spiking dynamics of ca1 pyramidal neurons. `https://senselab.med.yale.edu/ModelDB/showModel.cshtml?model=143719`, 2012.

[8] D. Bianchi, P. De Michele, C. Marchetti, B. Tirozzi, S. Cuomo, H. Marie, and M. Migliore. Effects of increasing creb-dependent transcription on the storage and recall processes in a hippocampal ca1 microcircuit. *Hippocampus*, 24(2):165–177, 2014. ISSN 1098–1063. doi: 10.1002/hipo.22212. URL `http://dx.doi.org/10.1002/hipo.22212`.

[9] D. Bianchi, P. De Michele, C. Marchetti, B. Tirozzi, S. Cuomo, H. Marie, and M. Migliore. Effects of increasing creb-dependent transcription on the storage and recall processes in a hippocampal ca1 microcircuit. `https://senselab.med.yale.edu/ModelDB/showModel.cshtml?model=151126`, 2014.

[10] R. Bitner. Cyclic amp response element-binding protein (creb) phosphorylation: a mechanistic marker in the development of memory enhancing alzheimer's disease therapeutics. *Biochem. Pharmacol.*, 83(6):705–714, 2012.

[11] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P.H. Goodman, F.C. Jr Harris, M. Zirpe, T. Natschlager, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, A.P. Muller, E. Davison, S. El Boustani, and A. Destexhe. Simulation of networks of spiking neurons: a review of tools and strategies. *J Comput Neurosci*, 23(3): 349–398, 2007.

[12] G. Buzsáki and E.I. Moser. Memory, navigation and theta rhythm in the hippocampal-entorhinal system. *Nat. Neurosci.*, 16(2):130–138, 2013.

[13] A. Chianese, F. Marulli, F. Piccialli, and I. Valente. A novel challenge into multimedia cultural heritage: An integrated approach to support cultural information enrichment. In *Proceedings - Signal-Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on*, pages 217–224, 2013.

[14] S. Cuomo, P. De Michele, and M. Chinnici. Parallel tools and techniques for biological cells modelling. *Buletinul Institutului Politehnic DIN IASI, Automatic Control and Computer Science Section*, LXI:61–75, 2011.

[15] S. Cuomo, P. De Michele, and R. Farina. A cublas-cuda implementation of pcg method of an ocean circulation model. In *Proceedings - Numerical Analysis and Applied Mathematics (ICNAAM), 9th International Conference of*, volume 1389, pages 1923–1926, 2011.

[16] S. Cuomo, P. De Michele, R. Farina, and M. Chinnici. Inverse preconditioning techniques on a gpus architecture in global ocean models. In *Proceedings - Recent Researches in Applied Mathematics and Informatics - Euro SIAM*, pages 15–20, 2011.

[17] S. Cuomo, P. De Michele, and R. Farina. An inverse preconditioner for a free surface ocean circulation model. In *AIP Conf. Proc. - Mathematical Problems in Engigneering, Aereospace and Sciences (ICNPAA), 9th International Conference on*, volume 1493, pages 356–362, 2012.

[18] S. Cuomo, P. De Michele, R. Farina, and F. Piccialli. A smart gpu implementation of an elliptic kernel for an ocean global circulation model. *Applied Mathematical Sciences*, 7(61):3007–3021, 2013.

[19] S. Cuomo, P. De Michele, and F. Piccialli. A regularized mri image reconstruction based on hessian penalty term on cpu/gpu systems. In *Proceedings - Computational Science (ICCS), International Conference on*, volume 18, pages 2643–2646, 2013.

[20] S. Cuomo, P. De Michele, and F. Piccialli. A performance evaluation of a parallel biological network microcircuit in neuron. *International Journal of Distributed & Parallel Systems*, 4(1):15–31, 2013.

[21] S. Cuomo, P. De Michele, A. Galletti, and G. Ponti. A biologically inspired model for analyzing behaviours in a social network community and cultural heritage scenario. In *Proceedings - Signal Image Technology & Internet based Systems (SITIS), The 10h International Conference on*, 2014.

[22] S. Cuomo, P. De Michele, and Posteraro M. A biologically inspired model for describing the user behaviors in a cultural heritage environment. In *Proceedings - Advanced Database Systems (SEBD), 22nd Italian Symposium on*, pages 292–302, 2014.

[23] S. Cuomo, P. De Michele, and F. Piccialli. 3d data denoising via nonlocal means filter by using parallel gpu strategies. *Comput Math Methods Med*, 2014.

[24] S. Cuomo, P. De Michele, G. Ponti, and Posteraro M. A clustering-based approach for a finest biological model generation describing visitor behaviours in a cultural heritage scenario. In *Proceedings - Data Management Technologies and Applications (DATA), 3rd International Conference on*, pages 427–433, 2014.

[25] V. Cutsuridis, S. Cobb, and B.P. Graham. Encoding and retrieval in a model of the hippocampal ca1 microcircuit. *Hippocampus*, 20:423–446, 2010.

[26] D. Debanne and M.M. Poo. Spike-timing dependent plasticity beyond synapse - pre- and post-synaptic plasticity of intrinsic neuronal excitability. *Front. Synaptic Neurosci.*, 2(21), 2010.

[27] Y. Dong, T. Green, D. Saal, H. Marie, R. Neve, E.J. Nestler, and R.C. Malenka. Creb modulates excitability of nucleus accumbens neurons. *Nat. Neurosci.*, 9: 475–477, 2006.

[28] H. Eichenbaum, P. Dunchenko, E. Wood, M. Shapiro, and H. Tanila. The hippocampus, memory and place cells: Is it spatial memory or a memory of space? *NEURON*, 23:209–226, 1999.

[29] T.F. Freund and G. Buzsaki. Interneurons of the hippocampus. *HIPPOCAMPUS*, 6:347–470, 1996.

[30] L. Fuentemilla, W.D. Penny, N. Cashdollar, N. Bunzeck, and E. Düzel. Theta-coupled periodic replay in working memory. *Curr. Biol.*, 20:606–612, 2010.

[31] Gasparini, S. and Migliore, M. and Magee, J.C. On the initiation and propagation of dendritic spikes in CA1 pyramidal neurons. *J Neurosci*, 24(49):11046–11056, 2004. doi: 10.1523/JNEUROSCI.2520-04.2004.

[32] N.L. Golding, N.P. Staff, and N. Spruston. Dendritic spikes as a mechanism for cooperative long-term potentiation. *Nature*, 418:326–331, 2002.

[33] B. Gong, O.V. Vitolo, F. Trinchese, S. Liu, M. Shelanski, and O. Arancio. Persistent improvement in synaptic and cognitive functions in an alzheimer mouse model following rolipram treatment. *J. Clin. Invest.*, 114:1624–1634, 2004.

[34] Graham, L. The Surf-Hippo Neuron Simulation System, v3.5, 2004. URL `http://www.neurophys.biomedicale.univ-paris5.fr/~{}graham/surf-hippo-files/Surf-Hippo.README.html`.

[35] P.L. Greer and M.E. Greenberg. From synapse to nucleus: calcium-dependent gene transcription in the control of synapse development and function. *Neuron*, 59(6):846–860, 2008.

[36] M.H. Han, C.A. Bolaños, T.A. Green, V.G. Olson, R.L. Neve, R.J. Liu, G.K. Aghajanian, and E.J. Nestler. Role of camp response element-binding protein in the rat locus ceruleus: regulation of neuronal activity and opiate withdrawal behaviors. *J. Neurosci.*, 26(17):4624–4629, 2006.

[37] J.A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

[38] M.E. Hasselmo, C. Bodelon, and B. Wyble. A proposed function of the hippocampal theta rhythm: Separate phases of encoding and retrieval of prior learning. *NEURAL COMPUT*, 14(4):793–817, 2002.

[39] M.L. Hines and N.T. Carnevale. The neuron simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[40] M.L. Hines and N.T. Carnevale. *The Handbook of Brain Theory and Neural Networks, 2nd ed.*, chapter The NEURON simulation environment, pages 769–773. MIT Press. Cambridge, 2003.

[41] M.L. Hines and N.T. Carnevale. Translating network models to parallel hardware in neuron. *J Neurosci Methods*, 169(2):425–455, 2008.

[42] M.L. Hines, H. Eichner, and F. Schurmann. Neuron splitting in compute-bound parallel network simulations enables runtime scaling with twice as many processors. *J Comput Neurosci*, 25:203–210, 2008.

[43] M.L. Hines, H. Markram, and F. Schurmann. Fully implicit parallel simulation of single neurons. *J Comput Neurosci*, 25(3):439–448, 2008.

[44] H. Hirase, X. Leinekugel, A. Czurkó, J. Csicsvari, and G. Buzsáki. Firing rates of hippocampal neurons are preserved during subsequent sleep episodes and modified by novel awake experience. *Proc. Natl. Acad. Sci. U. S. A.*, 98(16):9386–9390, 2001.

[45] A.L. Hodgkin and A.F. Huxley. Current carried by sodium and potassium ions through the membrane of the giant axon of loligo. *J. Physiol.*, 116:449–472, 1952.

[46] A.L. Hodgkin and A.F. Huxley. The components of membrane conductance in the gian axon of loligo. *J. Physiol.*, 116:473–496, 1952.

[47] A.L. Hodgkin and A.F. Huxley. The dual effect of membrane potential on sodium conductance in the gian axon of loligo. *J. Physiol.*, 116:497–506, 1952.

[48] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation nerve. *J. Physiol.*, 117:500–544, 1952.

[49] A.L. Hodgkin and W.A.H. Rushton. The electrical constants of a crustacean nerve fibre. *Proc. Roy. Soc.*, B-133:444–479, 1946.

[50] D.A. Hoffman, J.C. Magee, C.M. Colbert, and D. Johnston. K+ channel regulation of signal propagation in dendrites of hippocampal pyramidal neurons. *Nature*, 387 (6636):869–875, 1997.

[51] Y.H. Huang, Y. Lin, T.E. Brown, M.H. Han, D.B. Saal, R.L. Neve, R.S. Zukin, B.A. Sorg, E.J. Nestler, R.C. Malenka, and Y. Dong. Creb modulates the functional output of nucleus accumbens neurons: a critical role of n-methyl-d-aspartate glutamate receptor (nmdar) receptors. *J. Biol. Chem.*, 283:2751–2760, 2008.

[52] Q. Hui, W.M. Haddad, and J.M. Bailey. Multistability, bifurcations, and biological neural networks: A synaptic drive firing model for cerebral cortex transition in the induction of general anesthesia. *Nonlinear Analysis: Hybrid Systems*, 5(3):554–572, 2011.

[53] Q. Hui, W.M. Haddad, and J.M. Bailey. Multistability, bifurcations, and biological neural networks: A synaptic drive firing model for cerebral cortex transition in the induction of general anesthesia. *Nonlinear Analysis: Hybrid Systems*, 5:554–572, 2011.

[54] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[55] James M. Bower, David Beeman, et al. *The Book of GENESIS*. Internet Edition, 2003.

[56] E.R. Kandel. The molecular biology of memory storage: A dialogue between genes and synapses. *Science*, 294:1030–1038, 2001.

[57] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1):96–129, 1998.

[58] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[59] B.W. Kerninghan and R. Pike. *The UNIX Programming Environment*. Prentice-Hall Software Series, 1984.

[60] T. Klausberger, P.J. Magill, L.F. Marton, J. David, B. Roberts, P.M. Cobden, G. Buzsaki, and P. Somogyi. Brain-state- and cell-type-specific firing of hippocampal interneurons in vivo. *Nature*, 421:844–848, 2003.

[61] T. Klausberger, L.F. Marton, A. Baude, J.D. Roberts, P.J. Magill, and P. Somogyi. Spike timing of dendrite-targeting bistratified cells during hippocampal network oscillations in vivo. *Nat. Neurosci.*, 7:41–47, 2004.

[62] J. Kleinberg. The convergence of social and technological networks. *Commun. ACM*, 51(11):66–72, 2008.

[63] O. Kroupina and R. Rojas. A Survey of compartmental modelling packages. Technical Report B-04-08, Free University of Berlin, June 2004.

[64] R. Kumar, J. Novak, and A. Tomkins. *Link Mining: Models, Algorithms, and Applications*, chapter Structure and Evolution of Online Social Network, pages 337–357. J. Am. Soc. Inf. Sci. Technol., 2010.

[65] S. Káli and P. Dayan. The involvement of recurrent connections in area. ca3 in establishing the properties of place fields: a model. *J. Neurosci.*, 20:7463–7477, 2000.

[66] Lanza, G., Grossi, C.E., and Zaccheo, D. *L'organizzazione del sistema nervoso centrale*. Edi-ermes, 1989.

[67] L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907.

[68] L.S. Leung, L. Roth, and K.J. Canning. Entorhinal inputs to hippocampal ca1 and dentate gyrus in the rat: a current-source-density study. *J. Neurophysiol.*, 73: 2392–2403, 1995.

[69] J. Lisman. Working memory: the importance of theta and gamma oscillations. *Curr. Biol.*, 20:490–492, 2010.

[70] J.E. Lisman. Relating hippocampal circuitry to function: Recall of memory sequences by reciprocal dentate-ca3 interactions. *Neuron*, 22:233–242, 1999.

[71] R.J.A. Little and D.B. Rubin. *Statistical Analysis with Missing Data*. Wiley Editor, 2002.

[72] B.E. Lonze and D.D. Ginty. Function and regulation of creb family transcription factors in the nervous system. *Neuron*, 35:605–623, 2002.

[73] M. Lopez de Armentia, D. Jancic, R. Olivares, J.M. Alarcon, E.R. Kandel, and A. Barco. camp response element-binding protein-mediated gene expression increases the intrinsic excitability of ca1 pyramidal neurons. *J. Neurosci.*, 27:13909–13918, 2007.

[74] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. ISSN 0893-6080. doi: 10.1016/S0893-6080(97)00011-7. URL `http://www.sciencedirect.com/science/article/pii/S0893608097000117`.

[75] J.C. Magee. Dendritic hyperpolarization-activated currents modify the integrative properties of hippocampal ca1 pyramidal neurons. *J. Neurosci.*, 18(19):7613–7624, 1998.

[76] J.C. Magee and D. Johnston. Characterization of single voltage-gated na+ and ca2+ channels in apical dendrites of rat ca1 pyramidal neurons. *J. Physiol.*, 487.1: 67–90, 1995.

[77] H. Marie, W. Morishita, N. Calakos, and R.C. Malenka. Generation of silent synapses by acute in vivo expression of active camkiv or creb. *Neuron*, 45:741–752, 2005.

[78] W.S. McCulloch and W.H. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[79] Michael E. Hasselmo. The Role of Hippocampal Regions CA3 and CA1 in Matching Entorhinal Input With Retrieval of Associations Between Objects and Context: Theoretical Comment on Lee et al. (2005). *Behav Neurosci*, 119(1):342–345, 2005.

[80] S.E. Middleton, N.R. Shadbolt, and D.C. De Roure. Capturing interest through inference and visualization: Ontological user profiling in recommender systems. In *Proceedings - Knowledge Capture, 2Nd International Conference on*, pages 62–69, 2003.

[81] M. Migliore and G.M. Sheperd. An integrated approach to classifying neuronal phenotypes. *Nature Reviews Neuroscience*, 6:810–818, 2005.

[82] M. Migliore and G.M. Shepherd. Emerging rules for the distributions of active dendritic conductances. *Nat Rev Neurosci*, 3(5):362–370, 2002.

[83] M. Migliore and G.M. Shepherd. Functional significance of axonal kv7 channels in hippocampal pyramidal neurons. *Nat Rev Neurosci*, 6(10):810–818, 2005.

[84] M. Migliore, C. Cannia, W.W. Lyton, H. Markram, and M.L. Hines. Parallel network simulations with neuron. *J Comput Neurosci*, 21:119–129, 2006.

[85] B. Monti, C. Berteotti, and A. Contestabile. Subchronic rolipram delivery activates hippocampal creb and arc, enhances retention and slows down extinction of conditioned fear. *Neuropsychopharmacology*, 31:278–286, 2006.

[86] J.W. Moore and M. Hines. Simulations with neuron 3.1, on-line documentation in html format, available at, 1996. URL `http://neuron.duke.edu`.

[87] M. Nishiyama, K. Hong, K. Mikoshiba, M.M. Poo, and K. Kato. Calcium stores regulate the polarity and input specificity of synaptic modification. *Nature*, 408 (6812):584–588, 2000.

[88] N.T. Carnevale and M.L. Hines. *The NEURON book*. Cambridge University Press, 2006.

[89] L. Pagès, A. Gavaldà, and M.D. Lehner. Pde4 inhibitors: a review of current developments (2005 - 2009). *Expert. Opin. Ther. Pat.*, 19(11):1501–1519, 2009. doi: 10.1517/13543770903313753.

[90] G. Palma, F. Piccialli, P. De Michele, S. Cuomo, M. Comerci, P. Borrelli, and B. Alfano. 3d non-local means denoising via multi-gpu. In *Proceedings - Computer Science and Information Systems (FedCSIS), Federated Conference on*, pages 495–498, 2013.

[91] A.E. Pereda. Electrical synapses and their functional interactions with chemical synapses. *Nat Rev Neurosci*, 15(4):165–177, 2014. ISSN 250–263. doi: 10.1038/nrn3708. URL `http://dx.doi.org/10.1038/nrn3708`.

[92] P. Poirazi, T. Brannon, and B.W. Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37:989–999, 2003.

[93] D. Puzzo, O. Vitolo, F. Trinchese, J.P. Jacob, A. Palmeri, and O. Arancio. Amyloid-β-peptide inhibits activation of the nitricoxide/cgmp/creb pathway during hippocampal synaptic plasticity. *J. Neurosci.*, 25:6887–6897, 2005.

[94] W. Rall. Branching dendritic trees and motoneuron membrane resistivity. *Exp. Neurol.*, 1:491–527, 1959.

[95] W. Rall. Theory and physiological properties of dendrites. *Ann. N.Y. Acad. Sci.*, 96:1071–1092, 1962.

[96] L. Restivo, E. Tafi, M. Ammassari-Teule, and H. Marie. Viral-mediated expression of a constitutively active form of creb in hippocampal neurons increases memory. *Hippocampus*, 19:228–234, 2008.

[97] M.J. Sekeres, R.L. Neve, P.W. Frankland, and S.A. Josselyn. Dorsal hippocampal creb is both necessary and sufficient for spatial memory. *Learn. Mem.*, 17(6): 280–283, 2010.

[98] D.J Selkoe. Alzheimer's disease is a synaptic failure. *Science*, 298:789–791, 2002.

[99] M.M. Shah, M. Migliore, I. Valencia, E.C. Cooper, and D.A. Brown. Functional significance of axonal kv7 channels in hippocampal pyramidal neurons. *P Natl Acad Sci USA*, 105:7869–7874, 2008.

[100] Smith, R.G. NeuronC: a computational language for investigating functional architecture of neural circuits. *J NEUROSCI METH*, 43:83–108, 1992.

[101] A.F. Soleng, M. Raastad, and P. Andersen. Conduction latency along ca3 hippocampal axons from rat. *Hippocampus*, 13:953–961, 2003.

[102] P. Somogyi and T. Klausberger. Defined types of cortical interneurons structure space and spike timing in the hippocampus. *J PHYSIOL*, 562:9–26, 2005.

[103] B. Tirozzi, D. Bianchi, and E. Ferraro. *Introduction to Computational Neurobiology & Clustering.* World Scientific Book, 2007.

[104] A. Treves and E. Rolls. Computational analysis of the role of the hippocampus in memory. *HIPPOCAMPUS*, 4:374–391, 1994.

[105] T. Tully, R. Bourtchouladze, R. Scott, and J. Tallman. Targeting the creb pathway for memory enhancers. *Nat. Rev. Drug. Discov.*, 2:267–277, 2003.

[106] J. Viosca, M. Lopez de Armentia, D. Jancic, and A. Barco. Enhanced crebdependent gene expression increases the excitability of neurons in the basal amygdala and primes the consolidation of contextual and cued fear memory. *Learn Mem*, 16:193–197, 2009.

[107] J. Viosca, G. Malleret, R. Bourtchouladze, E. Benito, S. Vronskava, E.R. Kandel, and A. Barco. Chronic enhancement of creb activity in the hippocampus interferes with the retrieval of spatial information. *Learn Mem*, 16:198–209, 2009.

[108] J.R. Whitlock, A.J. Heynen, M.G. Shuler, and M.F. Bear. Learning induces long-term potentiation in the hippocampus. *Scince*, 313:1093–1097, 2006.

[109] E. Wood, P. Dunchenko, and H. Eichenbaum. The global record of memory in hippocampal neuronal activity. *NATURE*, 397:613–616, 1999.

[110] Wang X.J. and G. Buzsaki. Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model. *J. Neurosci.*, 16:6402–6413, 1996.

[111] A.P. Yiu, Rashid, A.J., and S.A. Josselyn. Increasing creb function in the ca1 region of dorsal hippocampus rescues the spatial memory deficits in a mouse model of alzheimer's disease. *Neuropsychopharmacology*, 36:2169–2186, 2011.

[112] Y. Zhou, J. Won, M.G. Karlsson, M. Zhou, T. Rogerson, J. Balaji, R. Neve, P. Poirazi, and A.J. Silva. Creb regulates excitability and the allocation of memory to subsets of neurons in the amygdala. *Nat. Neurosci.*, 12(11):1438–1443, 2009.