Statistical analysis of spike trains under variation of synaptic weights in neuronal networks

Gaia Lombardi

POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Matematica

STATISTICAL ANALYSIS OF SPIKE TRAINS UNDER VARIATION OF SYNAPTIC WEIGHTS IN NEURONAL NETWORKS

Relatore: Prof. Maurizio Verri
Correlatore: Prof. Bruno Cessac
Correlatore: Prof. Riccardo Sacco

Tesi di Laurea Magistrale di:
Gaia Lombardi
Matr. 769876

Anno Accademico 2012 - 2013

Contents

List of Figures

Sommario

Il progetto di tesi qui presentato è il risultato del lavoro svolto presso il Dipartimento di Neuroscienza Computazionale del centro di ricerca INRIA a Sophia Antipolis (Francia), nell'ambito di uno stage della durata di sei mesi, sotto la supervisione dei Professori Bruno Cessac e Pierre Kornprobst del gruppo di ricerca Neuromathcomp. Tale lavoro di ricerca è volto allo studio e al miglioramento di un'analisi statistica della distribuzione di probabilità di una sequenza di treni d'impulsi (potenziali d'azione) emessi dai neuroni in caso di stimoli provenienti dall'esterno. È stato ampiamente osservato sperimentalmente che i treni di impulsi registrati sono difficilmente riproducibili in un secondo esperimento anche sotto medesime condizioni, ma sono comunque parse evidenti delle regolarità statistiche. L'idea alla base di questo lavoro è di approssimare la distribuzione di probabilità "nascosta" dietro questi impulsi nervosi e di estrapolarla direttamente dai dati sperimentali.

Precedenti analisi di ricerca condotte da Cessac, in collaborazione con altri ricercatori, hanno permesso di sviluppare un modello statistico per i treni di impulsi basato sulla distribuzione di Gibbs per catene di Markov. Questa statistica è successivamente applicata ad un modello BMS (G. Belson, O. Mazet e H. Soula) che permette di modellizzare gli aspetti fisico-biologici della risposta neuronale pur mantenendo una limitata complessità di modellizzazione. Inoltre servendosi di questo modello neuronale per ricavare la statistica dei treni di impulsi è garantito il controllo della maggior parte dei parametri che influenzano l'attività del neurone stesso ed è stato così possibile introdurre nella formulazione la dipendenza temporale di alcuni di questi parametri.

Gran parte dei fisici e dei matematici nel settore della neuroscienza si trovano di fronte all'impegnativo obiettivo di giustificare matematicamente e riprodurre attraverso metodi numerici il ruolo delle sinapsi in diversi meccanismi cerebrali, come il processo di apprendimento o di memoria, e riconoscono il fondamentale ruolo che gioca il fattore temporale su questo tipo di processi e nell'elaborazione dell'informazione da parte dei neuroni. In tale contesto, questo lavoro di tesi rappresenta un sostanziale, seppur piccolo, passo in avanti verso il miglioramento e il potenziamento di un metodo di analisi

statistica dei treni d'impulsi nel modello BMS prendendo in considerazione la dipendenza temporale della risposta neuronale.

Basandosi sulla minimizzazione della divergenza di Kullback-Leibler che misura la "distanza" tra due misure di probabilità, sono stati impostati alcuni parametri del modello quali i pesi sinaptici e la corrente esterna. Per quel che riguarda l'implementazione e le simulazioni, questo metodo è stato sviluppato in C++ ed è stato testato sia su dati generati numericamente attraverso una rete di neuroni con il modello BMS, sia utilizzando veri dati sperimentali registrati da una retina in vitro.

Chiaramente, i risultati che si ottengono attraverso l'applicazione del metodo proposto su dati reali sono per il momento da considerarsi poco significativi per un'esaustiva interpretazione. Ciononostante, sembrano fornirci incoraggianti segnali che stimolano a proseguire in questa direzione e a utilizzare per esempio dati sperimentali di diversa natura o testare la stessa statistica su modelli neuronali alternativi a quello presentato in questo lavoro di tesi.

Abstract

The present work arises from a six month internship experience in the Computational Neuroscience Department with the Neuromathcomp team of research of INRIA institute in Sophia Antipolis (France), in particular with the major collaboration of professors Bruno Cessac and Pierre Kornprobst. It concerns an already existing statistical analysis of the hidden probability distribution of spike trains (or action potential) that neurons emit when they are subject to stimuli from the external world. In previous research activities conducted by Cessac and coworkers the spike trains statistics was applied to a neuron model, which seems to be a good compromise between biological characterization and modelistic tractability, where it is possible to control all the parameters involved and where we were able to easily add a new method to let the model be more flexible to parameter changes in time. Most of physicists and mathematicians who study neuroscience are dealing with the challenging purpose to explain mathematically and to reproduce numerically the role of synapses in brain mechanisms such as learning and memory, and it is widely recognized that the temporal component in neuronal processing information is an essential feature. In this context, the present work is a little step forward to improve the statistics of spike trains in a neuron model taking into account the temporal dependence of the neuron response. Based on the minimization of the Kullback-Leibler divergence, parameters of the neuron model such as synaptic weights and external input are adjusted. We first applied the method to data artificially generated with the neuron model and at the end of the internship to experimental data recorded from a real retina in vitro. Clearly, results from the application of the proposed method to the interpretation of real experimental data from a retina in vitro are still preliminary. Nevertheless, they seem to provide an encouraging indication to pursue testing the method with other experimental data of different nature or applying the method to other more complicated neuron models.

Chapter 1

Introduction

Neuroscience is a complex and amazing discipline aimed to explore the nervous system and to understand the biological basis of its behavior. The recent fast growing of this field inspired physicists and mathematicians to develop models permitting the analysis of the nervous system, where neurons communicate among them through electrical signals which carry information. However, basic questions are still open: how do they process, encode and transmit this information? The Neuromathcomp research team in INRIA (Institut National de Recherche en Informatique et en Automatique) located in Sophia Antipolis (France), has long-standing experience in the study neuron behavior and try to give answer to these and other questions from a mathematical and computational perspectives. They analyze the brain functions in terms of information processing properties of the neuron system and develop mathematical models for neuron network dynamics. In particular, they focused their studies on neurons placed in the retina, especially because of its interesting features. As a matter of fact, even though retina is an accessible part of the brain [23] and a prominent system to study the neurobiology and the computational capacity of coding signals, some of its neuronal functions are stil unknown.

The present thesis is part of an existing work in the statistical analysis of neuron activity in retina developed by B. Cessac and his colleagues [3], [37], [8], [5], [38], [4], [31]. This work focuses on methods from statistical physics and probability theory allowing the analysis of spike trains in neural networks. A spike train, as it will be explained in chapter 3, is a sequence of electrical signals that the neuron emits when it is subject to a stimulus from the external world and the role of the brain is to analyze this *neural*

code and to infer crucial information about the stimulus. In neuroscience how the brain can code these spike trains and how the stimulus is encoded by the nervous system are still unsolved questions. Moreover, spike trains are usually not reproducible when repeating the same experiment even with a very good control on stimuli or ensuring that the experimental conditions have not changed. Therefore, researchers are seeking statistical regularities in spike trains. The idea behind Cessac's work is to assume that the spike train statistics can be summarized by a hidden probability characterizing the probability of spiking patterns. Thus, one goal is to approximate this hidden probability distribution directly from experimental data, as shown in details in [6]. In chapter 3, we will describe some theoretical tools allowing to handle this purpose based on Cessac and Palacios' work [6]. It is worth recalling that since researchers are constantly seeking statistical regularities in spike trains, they define statistical indicators such as firing rate, probability of spike coincidence, spike response function, spike correlations and so on [1] [15] [17]. Approximating the probability distribution and providing an accurate model for spike train statistics is therefore an early step for *"reading the code"*.

Nevertheless, obtaining statistical models from experimental data or selecting a model among many others are difficult tasks. For example, it has been long believed that firing rates (the probability that a neuron emits a spike in a certain time interval) were carrying most of the information that neurons transmit. As a consequence the canonical probability distribution which reproduces the firing rates without additional assumptions, is a Bernoulli distribution, and the probability that a given number of spikes is emitted within a fixed time interval is a Poisson [20] [18]. However, more recent experiments evidenced the role of spike timing or spike synchronization, in processes such as vision or interactions between perception and motion [21] [29] [25]. Thus, one has to consider more elaborated statistical models for spike trains than Poisson distributions. Furthermore, extrapolating statistical models from experimental data, others problems occur such as obtaining "clean" data with a good control on parameters, dealing with finite sampling, non stationarity, synaptic plasticity or adaptation effects. Clearly, there is not only a viable solution to obtain statistical models and several approaches have been proposed in literature [32] [28].

As a consequence, it seems simpler to characterize statistics for spike trains in neuronal network models where it is possible to control exactly the parameters involved (number of neurons and samples and duration) [4] [3].

We will consider the BMS model (G. Belson, O. Mazet e H. Soula), a possible model for neurons which can be considered a good compromise between analytical tractability and biological realism. For sake of completeness in chapter 4 we will thoroughly discuss the BMS model (see [4] [3] for more details).

Another important aspect considered in this manuscript, which concerns the principal contribution of this thesis to the team of research, is the role of parameters time dependence in a neuron model. Nowadays in literature [2] [24] [12] [17] [16] [18] [19] [21] [22] [26], it is widely recognized that synapses, which are responsable for learning and memory, have a very relevant time dependent behavior (section 2.4). The temporal dependence is mostly due to synaptic plasticity (namely the ability of synapses to strengthen or weaken over time, in response to increases or decreases in their activity) which appears to play a central role in extrapolating the spike train statistics. The challenging purpose of this thesis is to add a temporal component in the existing theory carried on by B. Cessac. Minimizing the *Kullback-Leibler divergence* (sections 3.2.2, 4.4.5 and 4.5), which is in information theory [6] [34] [33] an indicator of the distance between two probability distributions (in our case the "real" one extrapolated from the neuron model and the hidden distribution we would like to find), we change over time two parameters of the model: synaptic weights and external input. We expect to find an approximating statistical distribution that permitts to improve these parameters of the neuron model such that the Kullback-Leibler divergence between this and the distribution we want to approximate goes to zero during simulation steps. Furthermore, we will apply our method, which is fully described in section 4.5 and 5.2, to experimental data recorded from a retina in vitro.

This work is organized as follows. In chapter 2 the anatomy of neurons and other basic features we will use along this manuscript are provided, in order to let the reader feel more comfortable with a not so spread field of research such as Neuroscience. We will continue in chapter 3 and 4 with statistical formalism and neuron models respectively, than with numerical implementation and simulations in chapters 5 and 6.

Chapter 2

Neuronal Anatomy

2.1 Neurons

A neuron is an electrically excitable cell that propagates signals rapidly, generating chemical and electrical pulses, called *action potentials* or *spikes*, which can travel down nerve fibers. Neurons represent and transmit information by firing sequences of spikes in various temporal patterns.

In practice, these electrical signals are the difference in electrical potential between the interior of a neuron and the surrounding extracellular medium. Under resting conditions, the potential inside the cell membrane has a value approximately of $-70mV$ with respect to the potential outside and the cell is said to be polarized. Ion pumps located in the cell membrane maintain concentration gradients that support this membrane potential difference. Ions, thus, flow into (and out of) a cell through open channels and create a current that makes the membrane potential more positive (or negative), a process called depolarization (or hyperpolarization). Then, if a neuron is depolarized sufficiently to raise the membrane potential above a threshold level the neuron immediately generates an action potential (or a spike). Action potentials play a centrale role in cell-to-cell communication, thanks to their speed and their ability to be propagated over large distances without attenuation.

A typical neuron (there is a wide variety in shape, size and properties) can be decomposed in three anatomical and functional parts, called *dendrites*, *soma* and *axon* as shown in figure 2.1. The *soma* is the central part, where the nucleus of the neuron is placed, and performs an important nonlinear processing activity. The *dendrites* are cellular extensions that allow the neuron to receive inputs from the other neurons through synaptic connections

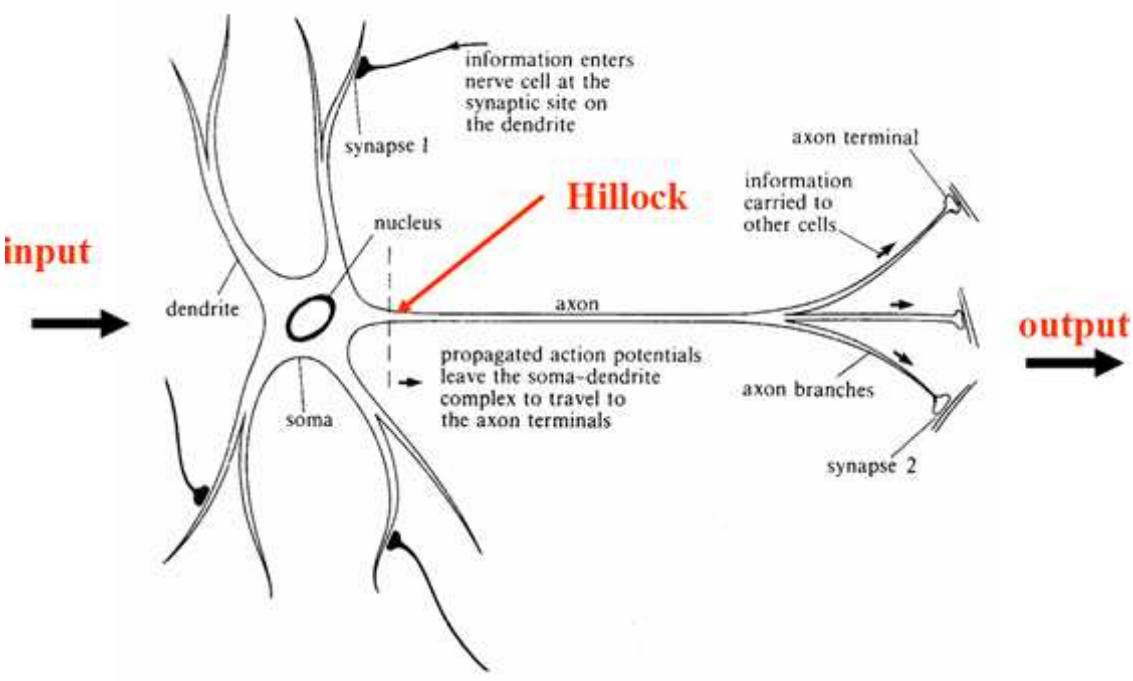


Figure 2.1: A neuron representation from www.kootation.com.

and that are considered as linear combiners of these inputs. The *axon* is a long projection of the nerve cell which carries nerve signals away from the soma (and sometimes carries information back to it). The *axon* is divided in three important parts: the axon hillock, the most easily-exited part of the neuron and the spike initiation zone for the axon; the central part of the axon which propagates the information and the axon terminal, which contains the synapses, specialized structures where neurotransmitter chemicals are released in order to communicate with target neurons.

2.2 Synapses

A synapse is a "connection" or junction between two neurons, the presynaptic and the postsynaptic neurons, that permits them respectively to send and to receive signals. There are two different kinds of synapses as shown in figure 2.2: the electrical and the chemical synapses. The first type uses gap junctions to connect the two neurons, namely this connection allows through ionic channels to transmit very quickly a change in membrane potential from one neuron to another. The chemical synapse, instead, uses neurotransmitter molecules, that are stored in "bags" called vesicles at the end of the axon. When a spike arrives, it causes the bags to fuse with the membrane of the cell and to release the neurotransmitters into the synaptic left (the gap between the two neuron connected with the synapse) in order to reach the "gates",*i.e.* ionic channels, of the second neuron and change the voltage inside. Thus, this second type of synapse involves complex mechanisms that start from a spike, an electrical pulse, that causes a chemical event that, in turn, causes an electrical change in the second neuron. Contrarily to electrical, a chemical synapse allows to control the way in which the second neuron is affected by the spike, by simply changing the number of ionic channels or the concentration of admitted ions.

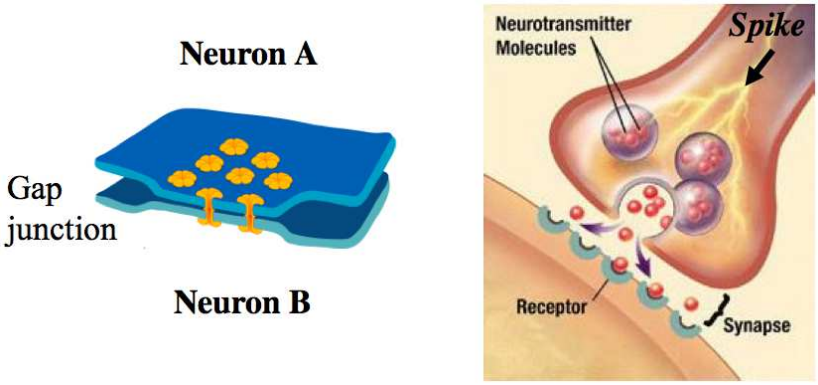


Figure 2.2: Types of synapses: Electrical and Chemical (from Rajesh P. N. Rao and Adrienne Fairhall's slides in Computational Neuroscience online course in by University of Washington)

The synapse can be either *excitatory* or *inhibitory*. An excitatory synapse tends to increase the postsynaptic membrane potential of the neuron that re-

ceives the signal (EPSP, excitatory postsynaptic potential). On the opposite, an inhibitory synapse tends to decrease the membrane potential (IPSP). The excitation usually happens thanks to sodium Na^+ and the inhibition thanks to potassium K^+. The first enters in the postsynaptic neuron, when the ionic channels are opened by the neurotransmitters, and causes *depolarization* in the cell, instead the potassium comes out from the neuron and causes *hyperpolarization*.
Synapses are responsable of *learning* and *memory* and regulate these activity through the *Synaptic Plasticity*.

2.3 Synaptic Plasticity

The *synaptic plasticity* is the ability of a synapse to change in strength the connection between neurons depending on how they participate in the firing activity. A clear example of Synaptic Plasticity is the *Hebbian Plasticity*, named after Donal O. Hebb (1904-1985) who was a Canadian psychologist, and it is explained as follows:

if a neuron A repeatedly takes part in firing neuron B, then the synapse from A to B is strengthened.

In brain experiments researchers have observed at least two main factors of evidence for Hebbian Plasticity: the LTP (long term potentiation) and the LTD (long term depression). It was experimentally observed in most of the areas of the brain, especially in the hippocampus, that when a neuron A is stimulated and this stimulation causes the neuron B to fire, it is possible to measure an increase in the EPSP from neuron A to neuron B that lasts for hours or days (LTP). The opposite (LTD) was observed if the neuron B does not fire when neuron A is stimulated.

2.4 From Hebbian rule to STDP

Moreover, it turns out that the synaptic plasticity depends on *Spike Timing*, that is the relative time of input and output spike. Namely when the postsynaptic neuron fires after the presynaptic neuron it was observed LTP, instead when the postsynaptic fires before there is an LTD effect. If we summarize this particular result for several intervals between input and output spikes, we get the learning windows, as shown in figure 2.3, that characterizes the synaptic plasticity. This effect is called STDP (spike time dependent plasticity) and it is a particular example of Hebbian learning.

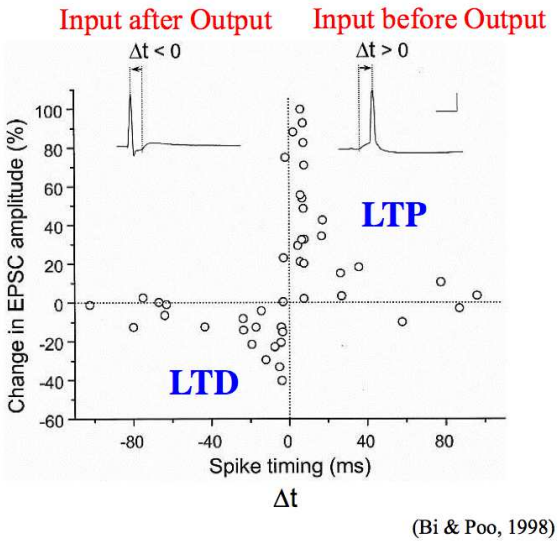


Figure 2.3: Learning Windows for synaptic plasticity (from Rajesh P. N. Rao and Adrienne Fairhall's slides in Computational Neuroscience online course in by University of Washington)

Chapter 3

Spike Trains Statistics

As discussed in the previous chapter, neurons when submitted to external stimuli or other neurons activity generate action potentials or "spikes", which are pulses of electrical signals. They respond to excitations by emitting sequences of spikes or "spike trains" and it is commonly believed that these spike trains carry information in their structure, *e.g.* spatial (neuron-dependent) and temporal (spike time) structure, and form a *neural code*. However, neural responses are not exactly reproducible: an experiment reproduced several times with the same conditions does not give rise to the same spike trains, although some regularity is observed. As a consequence, characterizing the relationship between stimuli and spike responses is equivalent to determining the most adequate probability distribution which relates a stimulus to its neural response. In mathematical terms, the problem of analyzing the neural code amounts to a problem of approximating from data some hidden probability measure μ of spiking patterns. There exist several attempts to infer this probability from data and/or general principles, based on Poisson or more general point processes, Bayesian approaches, Linear non-linear model [39], generalized linear model (GLM) [5] and many others. At the INRIA Department of NeuroMathComp for Neuroscience the presently most widely explored approach is the *Maximum Entropy Principle* [7] [6] [37] that allows to propose a spike train statistics model making restrictions to empirical observations. Before explaining this principle some definitions and notations [6] are useful to better understand the mathematical background of this write.

3.1 Definitions

3.1.1 Raster Plots

We consider a network of N neurons. We assume that there is a minimal time scale δ, set to $\delta = 1$ ms without loss of generality, namely there are no spikes between milliseconds, and than the dynamics can be discretized assuming the recording time as an integer. Each neuron i at each integer time t can be associated with a variable $\omega_i(t)$; $\omega_i(t)$=1 if neuron i fires at time t, $\omega_i(t)$=0 otherwise. A spiking pattern is a vector of the form $\omega(t) \stackrel{\text{def}}{=} [\omega_i(t)]_{i=1}^{N}$, and $\Omega = \{0,1\}^N$ is the set of spiking patterns. The $\omega_m^n = \omega(t)_{\{m \leq t \leq n\}}$ is a spike block of ordered list of spiking patterns in the set $\Omega = \{0,1\}^{N(n-m+1)}$.
We call a *raster plot* a bi-infinite sequence $\omega = \{\omega(t)\}_{t=-\infty}^{+\infty}$ of spiking patterns.

3.1.2 Transition Probabilities define Markov Chains

We are interested in the probability that neurons emit spikes at some times. Taking into account neuron dynamics where strong memory effects arise, it is likely that this probability depends on the entire history of the neuronal network and on many other biophysical parameters. A first possible approximation is to consider that the probability of a neuron firing at time t depends *only* on the spikes emitted in the past and not on the entire set of variables defining the neural activity (membrane potentials and so on) in the network. Then, we are seeking a family of transition probabilities of the form $P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right]$, where D is the memory depth of the evolution.
Transition probabilities with finite memory depth define a "*Markov Chain*", *i.e.* a random process where the probability to be in some state at time t depends only upon a finite past. Therefore, assume that we know the probability of occurence of the block ω_{t-D}^{t-1}

$$P\left[\omega_{t-D}^{t-1}\right] = P\left[\omega(t-D), \omega(t-D+1), \ldots, \omega(t-1)\right] \tag{3.1}$$

then, the probability of the block ω_{t-D}^{t} is

$$\begin{aligned} &P\left[\omega_{t-D}^{t}\right] = P\left[\omega(t-D), \omega(t-D+1), \ldots, \omega(t)\right] = \\ &P\left[\omega(t) \mid \omega(t-D), \omega(t-D+1), \ldots, \omega(t-1)\right] P\left[\omega(t-D), \omega(t-D+1), \ldots, \omega(t-1)\right] \end{aligned} \tag{3.2}$$

thus:

$$P\left[\omega_{t-D}^{t}\right] = P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right] P\left[\omega_{t-D}^{t-1}\right] \tag{3.3}$$

The main problem of this equation is that $P\left[\omega_{t-D}^{t}\right]$ depends on $P\left[\omega_{t-D}^{t-1}\right]$, which is not a *priori* determined and there are infinitely many choices for this value. Thus, a second assumptions is to consider time-invariant transistion probabilities. This means that for each possible spiking pattern $\alpha \in \Omega$, for all possible "memory" blocks $\alpha_{-D}^{-1} \in \Omega^D$ and $\forall t$, $P\left[\omega(t) = \alpha \mid \omega_{t-D}^{t-1} = \alpha_{-D}^{-1}\right] = P\left[\omega(0) = \alpha \mid \omega_{-D}^{-1} = \alpha_{-D}^{-1}\right]$. This is called a *stationary process*. In Markov Process theory this means to consider the irreducibility of the *Markov Chain*, *i.e.* $\exists t\ \forall i, j$ such that every transition from i to j is admissible, namely the strictly positivity of all probabilities. Moreover, if we assume the positivity together with the stationarity, the propriety of *ergodicity* of the system is ensured, or better it is ensured the existence of a unique probability μ, called the *asymptotic probability of the chain*, such that, it is possible to associate $P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right]$ with $\mu\left[\omega_{t-D}^{t-1}\right]$ in this way:

$$\mu\left[\omega_{t-D}^{t}\right] = \sum_{\omega(t)} P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right] \mu\left[\omega_{t-D}^{t-1}\right]. \tag{3.4}$$

Furthemore, the procedure can be iterated

$$\mu\left[\omega_{t-D}^{t}\right] = \sum_{\omega(t)} P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right] P\left[\omega(t-1) \mid \omega_{t-D}^{t-2}\right] \mu\left[\omega_{t-D}^{t-2}\right] \tag{3.5}$$

and so on. Thus, if the time interval $t - D$ is sufficiently long (tends to infinity) the probability of a block ω_{t-D}^{t} converges to $\mu\left[\omega_{t-D}^{t}\right]$ whatever $\mu\left[\omega_{t-D}^{t-1}\right]$ is. Namely, if the stationarity and the positivity of all probabilities hold, after a sufficiently long time it is possible to determine μ independently from the initial condition.

A fourth approximation we used in this report is the conditional independence between the N neurons and it reads:

$$P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right] = \prod_{i=1}^{N} P\left[\omega_i(t) \mid \omega_{t-D}^{t-1}\right] \tag{3.6}$$

3.1.3 Observables

We call *observable* a function which associates a real number to a raster plot

$$f : \Omega \longrightarrow \mathbb{R}. \tag{3.7}$$

An observable has a finite range k if:

$$f(\omega_0^{+\infty}) = f(\omega_0^{k-1}), \tag{3.8}$$

i.e. it depends only on the k first terms in the raster.
The average of f with respect to μ is :

$$\mu[f] = \int f d\mu. \tag{3.9}$$

3.1.4 Empirical averages

In experiments, raster plots have a finite duration T and one has only access to a finite number $\mathcal{N}$ of rasters denoted $\omega^{(1)}, ... , \omega^{(\mathcal{N})}$. From this data one computes *empirical averages* of observables. The first important assumption to compute empirical averages is the *stationarity* discussed in section 3.1.2 (time-invariant assumption). With stationarity it is possible to reduce the empirical average to a *time average*, $\pi_\omega^{(T)}[f]$ for an observable function f computed on the raster ω over time T.

$$\pi_\omega^{(T)}[f] = \frac{1}{T} \sum_{n=1}^{T} f(\omega_k(n)) \tag{3.10}$$

In addition, we know that for distinct rasters $\omega^{(1)}, \omega^{(2)}$, the empirical averages fluctuate from one to the other, *i.e.* $\pi_{\omega^{(1)}}^{(T)}[f] \neq \pi_{\omega^{(2)}}^{(T)}[f]$, and those fluctations depend on T. Anyway, maintaining the assumptions for the Markov Process of stationarity and ergodicity, the time averages converge almost surely to the hidden probability μ and we have

$$\pi_\omega^{(T)}[f] \underset{T \to +\infty}{\longrightarrow} \mu[f]. \tag{3.11}$$

3.1.5 Potential

A *Potential* is a function of the form

$$\begin{aligned} \phi_\beta : \qquad & \Omega \longrightarrow \mathbb{R}, \\ & \omega \longmapsto \sum_{k=1}^{N} \beta_k \mathcal{O}_k \end{aligned} \tag{3.12}$$

where the coefficients β_k are finite real numbers and $\mathcal{O}$ is an observable function of the form $\omega \longmapsto \prod_{u=1}^{r} \omega_{k_u}(n_u)$, where n_u is a time index and $u = 1, \ldots, r$ for some integer $r > 0$.
Moreover, the *Normalized Potential* for the distribution μ is the function:

$$\phi_t(\omega_{t-D}^t) \stackrel{\text{def}}{=} \log P\left[\omega(t)|\omega_{t-D}^{t-1}\right]. \tag{3.13}$$

3.1.6 Entropy

The entropy [9] is the maximal rate of information gain per time that can be achieved by coarse grained observations on a measure-preserving dynamical system and for a stationary probability distribution μ reads:

$$h[\mu] = -\lim_{n\to+\infty} \frac{1}{n} \sum_{\omega_1^n} \mu[\omega_1^n] \log \mu[\omega_1^n], \tag{3.14}$$

where the sum holds over all possible blocks ω_1^n. This definition holds for systems with infinite and finite memory. In the case of a Markov Chain with memory depth $D > 0$, we have

$$h[\mu] = -\sum_{\omega_1^{D+1}} \mu\left[\omega_1^D\right] P\left[\omega(D+1)|\omega_1^D\right] \log P\left[\omega(D+1)|\omega_1^D\right]. \tag{3.15}$$

and if $D = 0$

$$h[\mu] = -\sum_{\omega(0)} \mu\left[\omega(0)\right] \log \mu\left[\omega(0)\right]. \tag{3.16}$$

The entropy is zero if the probability is concentrated in one state and it reaches its maximum value if μ is equidistributed in Ω. For this reason, we can state that the entropy quantifies the information known about the probability μ.

3.2 Maximum Entropy Principle

Assume that the spike train statistics is distributed according to a hidden probability μ and we want to approximate this probability. Maximum entropy provides a method that allows us to approach μ from data. It selects among all probability distributions, consistent with empirical data constraints, the one with the highest entropy, *i.e.* the most random, to have a probability that includes the least amount of information we have about the system and no more. From the ergodic theory [10] we know that the probabilty maximizing the entropy is the *Gibbs distribution*.

3.2.1 Equilibrium distribution and the Gibbs Property

Suppose now to have finite number $\mathcal{N}$ of rasters that can be observed thanks to the potential ϕ_β, we are looking for a probability μ that maximizes the entropy among all distributions ν with a prescribed expected value $\nu[\phi_\beta] := \sum_{\omega^(r)\in\Omega} \nu[\omega]\phi_\beta(\omega)$ for the observable ϕ_β. This means we have to solve a variational problem under the constraint:

$$h[\mu] = \max\{h[\nu] : \nu[\phi_\beta] = E\}. \tag{3.17}$$

As the function $\nu \longmapsto h[\nu]$ is strictly concave, there is a unique maximising probability μ given the value E and it is obtained with the method of Lagrange multipliers [9]. The value E is extrapolated from experimental observations. Thus, the formula for the *topological pressure*

$$P(\phi_\beta) = \sup_{\nu\in\mathcal{M}_{inv}} (h[\nu] + \nu[\phi_\beta]) = h[\mu] + \mu[\phi_\beta], \tag{3.18}$$

where $\mathcal{M}_{inv}$ is the set of all possible stationary probabilities ν on the set of rasters with N neurons, maximizes the entropy given the information E we have about the system. Looking at the second equality, the variational principle selects among all possible probabilities ν one probability, which is the supremum, the *Gibbs distribution* μ. Moreover μ is the *equilibrium distribution* and for $D = 0$ (no memory) satisfies

$$\mu[\omega(0)] = \frac{e^{-\phi_\beta(\beta,\omega(0))}}{\sum_{\omega(0)} e^{-\phi_\beta(\beta,\omega(0))}}. \tag{3.19}$$

For a generic potential ϕ_β and more in generale for $D > 0$ it obeys: $\exists A, B > 0$ such that, for any block of the form ω_0^n, as in [5], holds

$$A \leq \frac{\mu[\omega_0^n]}{e^{-(n-D+1)P(\phi_\beta)}e^{\phi_\beta}} \leq B. \tag{3.20}$$

Once we have chosen a set of constraints and found the parameters β to maximize the entropy, how can we check the goodness of a model and moreover, if we would like to change the constraints how can we compare between two models?
A valid criterion for modeling comparison is the *Kullback-Leibler divergence* [33] [6].

3.2.2 Kullback-Leibler Divergence

Let μ, ν be two T-invariant measures. The Kullback-Leibler divergence between μ and ν is given by:

$$d(\mu, \nu) = \limsup_{n \to +\infty} \sum_{\omega_0^n} \mu[\omega_0^n] \log \left(\frac{\mu[\omega_0^n]}{\nu[\omega_0^n]} \right). \tag{3.21}$$

Minimizing the divergence corresponds to minimizing "what is not explained in the measure μ by the measure ν".
In our case where μ is the hidden probability measure, and is ergodic because of the assumptions in section 3.1.2, and μ_β is a Gibbs distribution with a potential ϕ_β, both defined in the same set of sequences, the following equality holds [6]:

$$d(\mu, \mu_{\phi_\beta}) = P(\phi_\beta) - \mu(\phi_\beta) - h(\mu). \tag{3.22}$$

This allows us to estimate the "distance" of our model from the hidden probability μ, providing the exact spike train statistics. The smaller $d(\mu, \mu_{\phi_\beta})$ the better is the model. Unfortunately, since μ is unknown this criterion looks useless. However, from section 3.1.4 we know that μ is well approximated by $\pi_\omega^{(T)}(\phi_\beta)$ which can be computed from the raster. Additionally, the entropy $h(\mu)$ is unknown. Anyway, if we consider two statistical models μ_{β_1}, μ_{β_2} with potential ϕ_{β_1}, ϕ_{β_2} to analyze the same data, it turns out that $h(\mu)$ is constant (because it only depends on data). Thus, comparing these two models is equal to compare $P(\phi_{\beta_1}) - \mu(\phi_{\beta_1})$ and $P(\phi_{\beta_1}) - \mu(\phi_{\beta_1})$. Finally,

the model ϕ_{β_1} is better than the model ϕ_{β_2} if

$$P(\phi_{\beta_2}) - \mu(\phi_{\beta_2}) \ll P(\phi_{\beta_1}) - \mu(\phi_{\beta_1}) \tag{3.23}$$

The choice of the potential (3.12), *i.e.* the choice of observables, gives the constraints for the statistical model. A normalization procedure allows to find the normalized potential from which the transition probabilities are constructed.

3.2.3 Transition matrix

Consider two spike blocks $\boldsymbol{\omega'}$, $\boldsymbol{\omega}$ of memory $D \leq 1$. The transition $\boldsymbol{\omega'} \longrightarrow \boldsymbol{\omega}$ is *legal* if $\boldsymbol{\omega'}$ has the form $\omega(t-D)\omega_{t-D+1}^{t-1}$ and $\boldsymbol{\omega}$ has the form $\omega_{t-D+1}^{t-1}\omega(t)$, *i.e.* $\boldsymbol{\omega'}$, $\boldsymbol{\omega}$ must correspond to overlapping blocks, for example:

$$\boldsymbol{\omega'} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}; \qquad \boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}. \tag{3.24}$$

Any block of the form ω_{t-D}^{t} of range $R = D+1$ can be viewed as a legal transition from the block $\boldsymbol{\omega'} = \omega_{t-D}^{t-1}$ to the block $\boldsymbol{\omega} = \omega_{t-D+1}^{t}$ and in this case we write $\omega_{t-D}^{t} \sim \boldsymbol{\omega'\omega}$.
The *transition matrix* is defined as:

$$\mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}} = \begin{cases} e^{\phi_\beta(\omega_{t-D}^{t})} & \text{if } \boldsymbol{\omega'\omega} \text{ is legal with } \omega_{t-D}^{t} \sim \boldsymbol{\omega'\omega} \\ 0, \text{ otherwise.} \end{cases} \tag{3.25}$$

This matrix has some properties useful to find the transition probabilities and than the normalized potential for solving the maximum entropy principle. Before examining these properties, we recall the *Perron-Frobenius theorem* for sake of completeness.

Theorem 1 (Perron-Frobenius Theorem)**.** *Let $\mathcal{L}$ be an irreducible matrix $M \times M$.*

1. *$\mathcal{L}$ has a positive (real) eigenvalue λ_{max} such that all other eigenvalues of $\mathcal{L}$ satisfy*
$$|\lambda| \leq \lambda_{max} \tag{3.26}$$

2. *Furthermore λ_{max} has algebric and geometric multiplicity one and has an eigenvector x such that $x > 0$.*

3. *Any non-negative eigenvector is a multiple of x.*

4. *More generally, if $y \geq 0,\ y \neq 0$ is a vector and μ is a number such that*
$$\mathcal{L}y \leq \mu y \tag{3.27}$$
then
$$y > 0,\ \textit{and}\ \mu \geq \lambda_{max} \tag{3.28}$$
with $\mu = \lambda_{max}$ if and only if is a multiple of x.

5. *If $0 \leq S \leq \mathcal{L},\ S \neq \mathcal{L}$ then every eigenvalue σ of S satisfy*
$$|\sigma| < \lambda_{max}. \tag{3.29}$$

6. *In particular, all the diagonal minors $\mathcal{L}_i$ obtained from $\mathcal{L}$ deleting the i-th row and column have eigenvalues all of which have absolute value λ_{max}.*

7. *If $\mathcal{L}$ is a primitive, then all other eigenvalues of $\mathcal{L}$ satisfy*
$$|\lambda| < \lambda_{max}. \tag{3.30}$$

3.2.4 Results from Perron-Frobenius Theorem

Let us first rewrite the transition matrix as follows:

$$\mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}} = \begin{cases} P\left[\omega(t) \mid \omega_{t-D}^{t-1}\right] & \text{if } \boldsymbol{\omega'} \sim \omega_{t-D}^{t-1}, \quad \boldsymbol{\omega} \sim \omega_{t-D}^{t} \\ 0, \text{ otherwise.} \end{cases} \tag{3.31}$$

So as Theorem 1 states, $\mathcal{L}$ has a unique eigenvalue λ_{max} real, positive and strictly larger than the modulus of all other eigenvalues.
It is possible to associate with this eigenvalue a left and a right eigenvectors l and r respectively, with strictly positive entries:

$$l\mathcal{L} = \lambda_{max} l, \qquad \mathcal{L} r = \lambda_{max} r. \tag{3.32}$$

Moreover λ_{max} obeys the bound

$$\min_{\boldsymbol{\omega'}} \sum_{\boldsymbol{\omega}} \mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}} \leq \lambda_{max} \leq \max_{\boldsymbol{\omega'}} \sum_{\boldsymbol{\omega}} \mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}}, \tag{3.33}$$

and using the constraint for the normalization of the conditional probabilities

$$\begin{aligned} &\sum_{\omega(t)} P\left[\omega(t) | \omega_{t-D}^{t-1}\right] = 1, \ \forall \omega_{t-D}^{t-1} \quad \Rightarrow \\ &\sum_{\boldsymbol{\omega}} \mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}} = 1, \ \forall \boldsymbol{\omega'}, \end{aligned} \tag{3.34}$$

it results $\lambda_{max} = 1$.
In addition, we have

$$\mathcal{L} r = r \quad \Leftrightarrow \quad \forall \boldsymbol{\omega'}, \ \sum_{\boldsymbol{\omega}} \mathcal{L}_{\boldsymbol{\omega'},\boldsymbol{\omega}} r_{\boldsymbol{\omega}} = \lambda_{max} r_{\boldsymbol{\omega'}} = r_{\boldsymbol{\omega'}}, \tag{3.35}$$

and $r_{\boldsymbol{\omega'}} = 1$ is a solution $\forall \boldsymbol{\omega}$.
Concerning the left eigenvector, l obeys

$$l\mathcal{L} = \lambda_{max} l = l. \tag{3.36}$$

Let $\langle l, r \rangle = \sum_{\boldsymbol{\omega'}} l_{\boldsymbol{\omega'}} r_{\boldsymbol{\omega'}}$ be the scalar product, then we have $\langle l, r \rangle = \sum_{\boldsymbol{\omega'}} l_{\boldsymbol{\omega'}}$.
Finally, setting

$$\mu_{\boldsymbol{\omega}} = \frac{l_{\boldsymbol{\omega}}}{\langle l, r \rangle} \tag{3.37}$$

we obtain that μ obeys:

- $\mu\mathcal{L} = \mu$;
- $\sum_{\boldsymbol{\omega}} \mu_{\boldsymbol{\omega}} = 1$.

These two relations express the fact that μ *is the unique invariant measure for the Markov Process* associated with the family of transition probabilities $P[\cdot|\cdot]$.

An additional consequence of *Perron-Frobenius Theorem* is the following convergence for all vector ν with positive entries:

$$\nu\mathcal{L}^n \underset{n\to+\infty}{\longrightarrow} \mu \tag{3.38}$$

where ν corresponds to a probability measure suitably normalized. This illustrates the convergence of the Markov Process and comes from the fact that the maximum eigenvalue is equal to one and the second dominates the others $\lambda_{max} = \lambda_1 = 1 \geq |\lambda_2| \geq |\lambda_3| ... \geq |\lambda_n| \geq 0$. Moreover, the convergence speed is exponential with a rate affected by the first and the second eigenvalues.
In term of spike statistics this means that, starting from any initial probability ν, $\nu\mathcal{L}^n$ which gives the probability of the block ω_{t-D}^{t-1} after n times steps converges to μ when $n \to +\infty$.

3.2.5 Remarks

The maximum entropy method, as well as other proposed in the literature, were mainly developed for data analysis: one speculates a form for transitions probabilities, performs parameter fitting and then uses the model to decode or to extrapolate the statistics of complex events. Clearly, obtaining statistical models from experimental data or selecting a model among many others are difficult tasks. One significant obstacle is the difficulty of obtaining "clean" data with a good control on parameter experiments, moreover one has to solve delicate questions such as the control of finite sampling effects (finite duration, finite number of experiments), extrapolation of the probability distribution characterizing small neural assemblies to a large population of neurons, non-stationarity, synaptic plasticity effects, and so on. It appears simpler to characterize spike train statistics in *neural networks models* where one controls exactly the neural network parameters, the number of involved neurons, the number of samples, and the duration of the experiment. So the questions now are: can we have a reasonable idea of what could be the spike train statistics studying a *neural network model*? Does Gibbs distribution arise in these models? What is the shape of the potential?
In the next chapter we will try to answer these questions providing first a more general idea of what a neural network model is.

Chapter 4

Neuron Models

4.1 Integrate-and-Fire model

We explained in chapter 2 that a neuron communicates with other neurons through pulses of electrical signals, called action potentials or spikes. During an action potential, the membrane potential follows a rapid and high trajectory (a pulse) and returns to a value that is hyperpolarized with respect to the threshold because of a chemical mechanism modeled quite accurately in literature. Nevertheless, neuron models can be simplified and simulations can be accelerated significantly if the chemical mechanisms are not explicitly included in the model. This is the main idea on which Integrate-and-Fire (IF) [14] [13] models are based, only considering neurons as points that emit a spike every time their membrane potential exceeds the threshold. By avoiding a biophysical description of the action potential, Integrate-and-Fire models are left with the simpler task of modeling only subthreshold membrane potential dynamics. This can be done with different level of rigor or details, depending on what the model is going to study. In this thesis we will analyze the statistics of a long sequence of spikes in a network of neurons, so it is useful to have a model for only one neuron simple to handle. A simpler version of IF model is the Leaky Integrate-and-fire (LIF) model.
The main idea is to consider the membrane cell and the system around it like an electric circuit with different components.

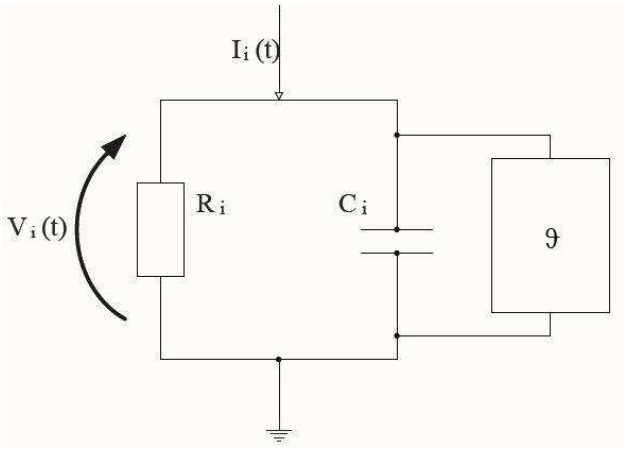


Figure 4.1: RC circuit

The Leak Integrate-and-Fire simplifies it with an RC circuit, which is studied applying the Kirchhoff's law and then Ohm's law in this way:

$$\begin{aligned} I_i(t) &= I_R + I_C, \\ I_i(t) &= g_i V_i(t) + C_i \frac{dV_i(t)}{dt}, \end{aligned} \tag{4.1}$$

where $g_i = \frac{1}{R_i}$ is the membrane conductance. In the more general form (the generalized Integrate-and-Fire model) g_i depends on V_i plus additional variables such as the probability of having ionic channels open and depends on time as well. Whereas for the Leak Integrate-and-Fire model, g_i is constant and $\tau_L = R_i C_i$ is the characteristic time for membrane potential decay when no current is present("the leak term").

$$\frac{dV_i}{dt} = -\frac{V_i}{\tau_L} + \frac{I_i(t)}{C_i}. \tag{4.2}$$

4.2 The BMS model

The *BMS model* was for the first time introduced by G. Belson, O. Mazet and H.Soula and it considers a neuron that emits a spike starting from the discrete time IF model. Indeed, if we consider that the Integrate-and-Fire will be used in numerical simulations, thus we can apply the Euler method in order to solve the differential equation. Fixing the sampling time $dt = 1$, the capacitance $C_i = 1$ and $\gamma = 1 - \frac{1}{\tau_L}$, where $\tau_L \geq 1$, thus $\gamma \in [0, 1[$, the equation reads:

$$\begin{aligned} &\frac{V_i(t+dt) - V_i(t)}{dt} = -\frac{V_i(t)}{\tau_L} + \frac{I_i(t)}{C_i}, \\ &V_i(t+1) = V_i(t) - \frac{V_i(t)}{\tau_L} + \frac{I_i(t)}{C_i}, \\ &V_i(t+1) = (1 - \frac{1}{\tau_L})V_i(t) + \frac{I_i(t)}{C_i}, \\ &V_i(t+1) = \gamma V_i(t) + \frac{I_i(t)}{C_i}, \\ &V_i(t+1) = \gamma V_i(t) + I_i(t). \end{aligned} \tag{4.3}$$

The synaptic weights are taken into account in the $N \times N$ matrix $\boldsymbol{W}$, that has entries W_{ij}, the weight of connection from neuron j to neuron i. The synaptic effect is incorporated in the input current, that reads

$$I_i(t) = \sum_{j=1}^{N} W_{ij} Z[V_j] + I_i, \tag{4.4}$$

where I_i is an external input applied to neuron i that does not depend on time. Moreover, a spike is modeled by the function $Z(x) = \chi(x) \geq \theta$ where $\chi(x)$ is equal to 1 when $x > \theta$ and 0 otherwise. In this context the BMS model reads

$$\begin{aligned} &\boldsymbol{V}(t+1) = \boldsymbol{F}(\boldsymbol{V}(t)), \\ &F_i(\boldsymbol{V}) = \gamma V_i(1 - Z[V_i]) + \sum_{j=1}^{N} W_{ij} Z[V_j] + I_i; \qquad i = 1, \dots, N. \end{aligned} \tag{4.5}$$

where $\boldsymbol{V} = [V_i]_{i=1}^{N}$ is the vector of the membrane potentials. In order to make dynamics more general and realistic we add a stochastic part in the equation

above:

$$\begin{aligned} &\boldsymbol{V}(t+1) = \boldsymbol{F}(\boldsymbol{V}(t)) + \sigma_B \boldsymbol{B}(t), \\ &F_i(\boldsymbol{V}) = \gamma V_i(1 - Z[V_i]) + \sum_{j=1}^{N} W_{ij} Z[V_j] + I_i; \qquad i = 1, \dots, N. \end{aligned} \tag{4.6}$$

where $\boldsymbol{B}(t) = [B_i]_{i=1}^N$ is an additive noise with Gaussian identically distributed and independent entries $B_i(t)$ with zero mean and variance 1 and σ_B is the noise amplitude. From a biophysical point of view, the noise term can be interpreted as the random variation in the ionic flux of charges crossing the membrane per unit time and it gives to it more general features.
Here a spike is modeled by the function Z and after that the membrane potential is reset instantaneously to a value V_{reset}, corresponding to the value of the membrane potential when the neuron is at rest (in this case it is *zero*).
Considering a network of neurons for this model, each neuron i receives input from other neurons (coming from the synaptic part in the equation (4.6)) and from the environment (the constant external current and the noise in equation (4.6)). If a neuron does not fire and does not receive influences from other neurons or input, then the membrane potential decays exponentially fast with a decay rate $0 < \gamma < 1$.

4.3 Markov chain for the BMS model

4.3.1 The last firing time

For $(s,t) \in \mathbb{Z}^2, s < t$, and each $i = 1, \dots, N$, we define now the *last firing time* of neuron i in the sequence ω_s^t as:

$$\tau_i(\omega_s^t) \stackrel{\text{def}}{=} \begin{cases} s, \quad \text{if } \ \omega_i(k) = 0, \quad k = s, \dots, t; \\ \max\{s \le k \le t, \ \omega_i(k) = 1\}, \quad \text{if } \ \exists k \in \{s, \dots, t\} \quad \text{such that } \ \omega_i(k) = 1 \end{cases} \tag{4.7}$$

Therefore, $\tau_i(\omega_s^t) = s$ either if neuron i fires at time s or if it does not fire during the whole time interval $[s,t]$.

4.3.2 The conditional probability for the membrane potential

We are interested in the conditional probability $P[\omega(t+1)|\omega_s^t]$ that, subject to appropriate assumptions as shown in section 1.3, is a variable length Markov Chain. As explained above, a neuron fires when its membrane potential exceeds the threshold θ, so

$$P[\omega_i(t+1)|\omega_s^t] = \omega_i(t+1)P[V_i(t+1) \geq \theta|\omega_s^t] + (1-\omega_i(t+1))P[V_i(t+1) \leq \theta|\omega_s^t]. \tag{4.8}$$

Analyzing the equation (4.6) we can show [3] that for each $(s,t) \in \mathbb{Z}^2, s < t$, conditionally to $Z(V_s^t) = \omega_s^t$ and given $V(s)$ the initial condition,

$$V_i(t+1) = \begin{cases} \gamma^{t+1-s}V_i(s) + C_i(\omega_s^t) + \sigma_B\xi_i(\omega_s^t) & \text{if neuron } i \text{ didn't fire} \\ & \text{in the interval [s,t]} \\ C_i(\omega_s^t) + \sigma_B\xi_i(\omega_s^t) & \text{otherwise.} \end{cases} \tag{4.9}$$

where:

$$\begin{aligned} C_i(\omega_s^t) &= \sum_{j=1}^{N} W_{ij}x_{ij}(\omega_s^t) + I_i\frac{1-\gamma^{t+1-\tau_i(\omega_s^t)}}{1-\gamma} \\ x_{ij}(\omega_s^t) &= \sum_{l=\tau_i(\omega_s^t)}^{t} \gamma^{t-l}\omega_j(l) \\ \xi_i(\omega_s^t) &= \sum_{l=\tau_i(\omega_s^t)}^{t} \gamma^{t-l}B_i(l). \end{aligned} \tag{4.10}$$

Clearly, the membrane potential $V_i(t+1)$ is the sum of a "deterministic" part $\gamma^{t+1-s}V_i(s) + C_i(\omega_s^t)$, and a stochastic part, $\sigma_B\xi_i(\omega_s^t)$. This term is fixed by the spike sequence ω_s^t and the $\xi_i(\omega_s^t)$'s are Gaussian, independent with mean zero and variance $\frac{1-\gamma^{2(t+1-\tau_i(\omega_s^t))}}{1-\gamma^2}$ because it is a finite sum of Normal Gaussian B_i.

Thus, the membrane potential $V_i(t+1)$ is Gaussian with mean

$$E[V_i(t+1)|\omega_s^t, V(s)] = \begin{cases} \gamma^{t+1-s}V_i(s) + C_i(\omega_s^t) & \text{if neuron } i \text{ didn't fire} \\ & \text{in the interval [s,t]} \\ C_i(\omega_s^t) & \text{otherwise.} \end{cases} \tag{4.11}$$

and covariance:

$$Cov[V_i(t+1), V_j(t+1)|\omega_s^t, V(s)] = \sigma_i^2(\omega_s^t)\delta_{ij}. \tag{4.12}$$

with:

$$\sigma_i^2(\omega_s^t) = \sigma_B^2 \frac{1-\gamma^{2(t+1-\tau_i(\omega_s^t))}}{1-\gamma^2}. \tag{4.13}$$

It is easy to show [3] that the $V_i(t+1)$'s are conditionally independent.

Further, the probability that a neuron does not fire within the interval $(s,t) \in \mathbb{Z}^2$ is given by:

$$P\left[\bigcap_{n=s}^{t}\{V_i(n) < \theta\}\right] = \sum_{\omega_s^t \in \Omega^{t-s}} P\left[\bigcap_{n=s}^{t}\{V_i(n) < \theta\}|\omega_s^t\right] P\left[\omega_s^t\right] =$$

$$= \sum_{\omega_s^t \in \Omega^{t-s}} \prod_{n=s+1}^{t} P\left[\{V_i(n) < \theta\}|\bigcap_{l=s}^{n-1}\{V_i(l) < \theta\} \cap \omega_s^{n-1}\right] P\left[\{V_i(s) < \theta|\omega_s^s\}\right] P\left[\omega_s^t\right]. \tag{4.14}$$

Since, in this case, $\xi_i(\omega_s^t)$ is Gaussian, centered, with variance $\frac{1-\gamma^{2(n-s)}}{1-\gamma^2}$ we have:

$$P\left[\{V_i(n) < \theta\}|\bigcap_{l=s}^{n-1}\{V_i(l) < \theta\} \cap \omega_s^{n-1}\right] =$$

$$= P\left[\gamma^{n-s}V_i(s) + C_i(\omega_s^{n-1}) + \sigma_B\xi_i(\omega_s^{n-1}) < \theta\right] = 1 - \Pi\left(\frac{\theta - \gamma^{n-s}V_i(s) - C_i(\omega_s^{n-1})}{\sigma_B\sqrt{\frac{1-\gamma^{2(n-s)}}{1-\gamma^2}}}\right) \tag{4.15}$$

where $\Pi(x) = \frac{1}{\sqrt{2\pi}}\int_x^{+\infty} e^{-\frac{u^2}{2}}\,du$.

4.3.3 Some remarks

A first remark is to underline that equation (4.11) expresses the loss of memory of a neuron whenever it fires. This is due to the reset of the membrane potential after firing and simplifies the following analysis.

Another significant feature to consider is the composition of the membrane potential. Indeed, it is the sum of a "deterministic" part, $\gamma^{t+1-s}V_i(s) + C_i(\omega_s^t)$, fixed by the initial condition at time s and by the spike sequence ω_s^t, and a stochastic part, $\sigma_B\xi_i(\omega_s^t)$, where the probability distribution of the noise $\xi_i(\omega_s^t)$ is also fixed by the spike sequence ω_s^t. More precisely, if B_i's are independent Gaussian with mean zero and variance 1, the $\xi_i(\omega_s^t)$'s are Gaussian independent with mean zero and variance $\frac{1-\gamma^{2(t+1-\tau_i(\omega_s^t))}}{1-\gamma^2}$.

The dependence on the initial condition $V(s)$ could be an obstacle to solve the dynamic of membrane potential because it is not known a *priori*. It is important to underline that when considering the evolution of a set of neurons, one starts from some "initial" time s. This time corresponds to the beginning of the experiment, but not with the beginning of the system under study which has undergone a previous evolution that actually determines the distribution of membrane potentials at time s. Therefore, to compute the distribution of membrane potentials at time s one has to consider the previous evolution of the system, which only postpones the problem, unless one assumes that this initial condition was drawn in an *infinite past*. An infinite past means a quite longer time than all characteristic time scales in the system, that can be justify mathematically with the asymptotic dynamic (infinite!!). In this case the initial condition is fixed in the infinite past, *i.e* $s \to -\infty$.
In this case $\gamma^{t+1-s}V_i(s) \to 0$ and, considering the sequence $\omega_{-\infty}^t$ the last firing time becomes:

$$\tau_i(\omega_{-\infty}^t) \stackrel{\text{def}}{=} \begin{cases} -\infty, \quad \text{if} \;\; \omega_i(k) = 0, \quad k = s, \ldots, t; \\ \max\left\{-\infty \leq k \leq t, \;\; \omega_i(k) = 1\right\}, \quad \text{if} \;\; \exists k \in \{-\infty, \ldots, t\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{such that} \;\; \omega_i(k) = 1. \end{cases} \tag{4.16}$$

Therefore the mean of $V_i(t+1)$ for each $t \in \mathbb{Z}$, conditionally to $\omega_{-\infty}^t$ is only equal to $C_i(\omega_s^t)$.

4.3.4 The transition probability

We finally define the transition probability for the BMS model as follows:

$$P\left[\omega(t+1)|\omega_{-\infty}^{t}\right] = \prod_{i=1}^{N} P\left[\omega_i(t+1)|\omega_{-\infty}^{t}\right] =$$

$$= \prod_{i=1}^{N} \left[\omega_i(t+1) P\left[\{V_i(s) \leq \theta|\omega_{-\infty}^{t}\}\right] + (1-\omega_i(t+1))\, P\left[\{V_i(s) < \theta|\omega_{-\infty}^{t}\}\right]\right] =$$

$$= \prod_{i=1}^{N} \left[\omega_i(t+1)\Pi\left(\frac{\theta - C_i(\omega_{-\infty}^{t})}{\sigma_i(\omega_{-\infty}^{t})}\right) + (1-\omega_i(t+1))\left(1-\Pi\left(\frac{\theta - C_i(\omega_{-\infty}^{t})}{\sigma_i(\omega_{-\infty}^{t})}\right)\right)\right]. \tag{4.17}$$

In these formulas $P\left[\omega(t+1)|\omega_{-\infty}^{t}\right]$ acts as a transition probability, as in Markov chains, but we have to pay attention because there is an infinite past dependence. Fortunately, the length of the Markov chain depends on the last firing time of each neuron, so we can state that:

$$P\left[\omega(t+1)|\omega_{-\infty}^{t}\right] = P\left[\omega(t+1)|\omega_{\tau(\omega_{-\infty}^{t})}^{t}\right] \tag{4.18}$$

The only obstacle here is that we cannot bound $\tau(\omega_{-\infty}^{t})$ even if this time is almost-surely finite. So we have to consider a process where transition probability may have an unbounded memory. This type of process is called "variable length Markov chain".

4.3.5 Stationarity

For the BMS model under the assumptions considered so far, it is possible to prove directly the stationarity of the Markov chain that reads:

$$\begin{aligned} &\text{Fix a sequence } a_{-\infty}^{0},\ a(-n) \in \Omega,\ n \leq 0,\quad \forall t \in \mathbb{Z} \\ &P[\omega(t) = a(0) | \omega_{-\infty}^{t-1} = a_{-\infty}^{-1}] = P[\omega(0) = a(0) | \omega_{-\infty}^{-1} = a_{-\infty}^{-1}] \end{aligned} \tag{4.19}$$

Assuming that $\omega(t-n) = a(-n),\ n \geq 0$ we have $\tau(\omega_{-\infty}^{t-1}) = t + \tau(a_{-\infty}^{-1})$ so, according to equation (4.10),

$$\begin{aligned} x_{ij}(\omega_{-\infty}^{t-1}) = \sum_{l=\tau_i(\omega_{-\infty}^{t-1})}^{t-1} \gamma^{t-1-l} \omega_j(l) = \sum_{l=t+\tau_i(a_{-\infty}^{-1})}^{t-1} \gamma^{t-1-l} a_j(l-t) = \\ = \sum_{l'=\tau_i(a_{-\infty}^{-1})}^{-1} \gamma^{-1-l'} a_j(l') = x_{ij}(a_{-\infty}^{-1}) \end{aligned} \tag{4.20}$$

and

$$\begin{aligned} C_i(\omega_{-\infty}^{t-1}) = \sum_{j=1}^{N} W_{ij} x_{ij}(\omega_{-\infty}^{t-1}) + I_i \frac{1 - \gamma^{t - \tau_i(\omega_{-\infty}^{t-1})}}{1-\gamma} = \\ \sum_{j=1}^{N} W_{ij} x_{ij}(a_{-\infty}^{-1}) + I_i \frac{1 - \gamma^{-\tau_i(\omega_{-\infty}^{-1})}}{1-\gamma} = C_i(a_{-\infty}^{-1}) \end{aligned} \tag{4.21}$$

Note that this last propriety holds because I_i does not depend on time. We have also

$$\sigma_i^2(\omega_{-\infty}^{t-1}) = \sigma_i^2(a_{-\infty}^{-1}). \tag{4.22}$$

Finally, we obtain (4.19) from:

$$\begin{aligned} &P[\omega(t) = a(0) | \omega_{-\infty}^{t-1} = a_{-\infty}^{-1}] = \\ &= \prod_{i=1}^{N} \left[\omega_i(t) \Pi \left(\frac{\theta - C_i(\omega_{-\infty}^{t-1})}{\sigma_i(\omega_{-\infty}^{t-1})} \right) + (1 - \omega_i(t)) \left(1 - \Pi \left(\frac{\theta - C_i(\omega_{-\infty}^{t-1})}{\sigma_i(\omega_{-\infty}^{t-1})} \right) \right) \right] = \\ &= \prod_{i=1}^{N} \left[a_i(0) \Pi \left(\frac{\theta - C_i(a_{-\infty}^{-1})}{\sigma_i(a_{-\infty}^{-1})} \right) + (1 - a_i(0)) \left(1 - \Pi \left(\frac{\theta - C_i(a_{-\infty}^{-1})}{\sigma_i(a_{-\infty}^{-1})} \right) \right) \right] = \\ &= P[\omega(0) = a(0) | \omega_{-\infty}^{-1} = a_{-\infty}^{-1}]. \end{aligned} \tag{4.23}$$

From stationarity we can restrict the analysis to transition probabilities of the form $P[\omega(0)|\omega_{-\infty}^{-1}]$. From now on we define, for the sake of simplicity, $\omega = \omega_{-\infty}^{0}$, $\underline{\omega} = \omega_{-\infty}^{-1}$ ($\underline{\omega}$ is called an "history"). Moreover, call T the *right shift* over X, *i.e* $(T\omega)(t) = \omega(t-1),\ t \geq 0$, we note that ωa is the right concatenation of ω and $a \in \Omega$, namely this is the sequence ω' such that $\omega'(t-1) = \omega(t),\ t \geq 0$ and $\omega'(0) = a$. Finally note that $T(\omega a) = \omega$.
In [3] it is proved that the system (4.6) admits a unique invariant probability measure that satisfies a variational principle (equilibrium state) and has the form of a Gibbs distribution.

4.4 Maximum entropy principle applied to the BMS model

4.4.1 Equilibrium state

Let $\psi : X \longrightarrow \mathbb{R}$ be a continuous function such that

$$\sum_{k=0}^{\infty} var_k(\psi) < \infty, \tag{4.24}$$

where $var_k(\psi)$ is the variation of the function ψ:

$$var_k(\psi) = \sup\{|\psi(\omega) - \psi(\omega')| : \omega, \omega' \in X,\ \omega(t) = \omega'(t),\ \forall t \in \{-k,\ \ldots, 0\}\}. \tag{4.25}$$

For μ invariant measure, we define the entropy as we did in section 3.1.6 with a potential ψ which obeys (3.12).

An *equilibrium state* μ_ψ is an invariant measure [10], such that the topological pressure $P(\psi)$ holds and it is zero whenever the potential is normalized as follows:

$$\psi(\omega) = \log P[\omega(0)|\underline{\omega}] \tag{4.26}$$

and we can write

$$\psi(\omega) \equiv \psi(\omega_{-\infty}^{0}) = \log P[\omega(0)|\underline{\omega}] =$$
$$= \sum_{i=1}^{N} \left[\omega_i(0) \log \left[\Pi \left(\frac{\theta - C_i(\underline{\omega})}{\sigma_i(\underline{\omega})} \right) \right] + (1 - \omega_i(0)) \log \left[1 - \Pi \left(\frac{\theta - C_i(\underline{\omega})}{\sigma_i(\underline{\omega})} \right) \right] \right]. \tag{4.27}$$

4.4.2 Entropy for the BMS model

We know that for the normalized potential (4.26) the topological pressure is zero $0 = h[\mu_\psi] + \mu_\psi[\psi]$. Therefore, the entropy reads:

$$h[\mu_\psi] =$$
$$= -\sum_{i=1}^{N} \mu_\psi \left(\left[\omega_i(0) \log \left[\Pi \left(\frac{\theta - C_i(\underline{\omega})}{\sigma_i(\underline{\omega})} \right) \right] + (1 - \omega_i(0)) \log \left[1 - \Pi \left(\frac{\theta - C_i(\underline{\omega})}{\sigma_i(\underline{\omega})} \right) \right] \right] \right). \tag{4.28}$$

It is important to underline that the potential is always negative. Thus, the entropy is positive whatever the value of parameters $W_{ij},\ N,\ \theta,\ \gamma$.

4.4.3 Finite Range approximation

The main difficulty in handling the transition probabilities and equilibrium state is the dependence on an history, dating back to $\tau(\omega_{-\infty}^{0})$ that is not bounded. Nevertheless, what we know is that the influence of history network on the membrane potential V_i at time t appears in the term $x_{ij}(\omega_{-\infty}^{0}) = \sum_{l=\tau_i(\omega_{-\infty}^{0})}^{0} \gamma^{-l} \omega_j(l)$ that for $l \to -\infty$ decays esponenatially fast because of the leak term γ [3]. Thus, one may argue that after a characteristic time depending on $\frac{1}{\log(\gamma)}$ the past network activity has a very small influence on $V_i(0)$. We can truncate the history with a finite time horizon R such that the membrane potential $V_i(0)$ depends on the past only up to finite time $-R$. In this setting the transition probability $P[\omega(0)|\omega_{-\infty}^{-1}]$ is approximated with $P[\omega(0)|\omega_{-R}^{-1}]$, where the memory is limited to at most R time steps in the past. These approximated transition probabilities constitute therefore a Markov chain with a memory depth R. Now the question is: how good is this approximation?

4.4.4 Convergence of the finite approximation

We define the range-$R+1$ potential as

$$\psi^{(R)}(\omega_{-R}^{0}) = \psi^{(R)}(\omega) = \\ \sum_{i=1}^{N}\left[\omega_i(0)\log\left[\Pi\left(\frac{\theta - C_i(\omega_{-R}^{-1})}{\sigma_i(\omega_{-R}^{-1})}\right)\right] + (1-\omega_i(0))\log\left[1-\Pi\left(\frac{\theta - C_i(\omega_{-R}^{-1})}{\sigma_i(\omega_{-R}^{-1})}\right)\right]\right], \tag{4.29}$$

and we compare how "distant" this potential is from the infinite range potential (4.26), evaluating the sup norm as follows:

$$\left\|\psi - \psi^{(R)}\right\|_{\infty} \leq \sup\{|\psi(\omega) - \psi(\omega')| : \omega, \omega' \in X,\ \omega(t) = \omega'(t),\ \forall t \in \{-R,\ \ldots, 0\}\} \\ \stackrel{\text{def}}{=} var_R(\psi) \tag{4.30}$$

which, it is possible to prove (see details in [3]), is bounded and we have

$$\left\|\psi - \psi^{(R)}\right\|_{\infty} \leq K\gamma^R \tag{4.31}$$

where K is an suitable constant, found in details in [3].

To compare the distance between the Gibbs distributions μ_ψ and $\mu_{\psi^{(R)}}$, we need the Kullback-Leibler divergence introduced in section 3.2.2, which is a criterion to provide some notion of asymmetric "distance" between two probabilities.

In the case where μ is an ergodic measure and μ_ψ a Gibbs state with a potential ψ, both defined in the same set of sequences, from (4.36) we obtain:

$$d(\mu, \mu_\psi) = P(\psi) - \mu(\psi) - h(\mu). \tag{4.32}$$

Additionally, the potential is normalized so $P(\psi) = 0$ and:

$$d(\mu_{\psi^{(R)}}, \mu_\psi) = P(\psi) - \mu_{\psi^{(R)}}(\psi) - h(\mu_{\psi^{(R)}}) = \mu_{\psi^{(R)}}(\psi^{(R)} - \psi), \tag{4.33}$$

where $h(\mu_{\psi^{(R)}}) = -\mu_{\psi^{(R)}}(\psi^{(R)})$.

Therefore,

$$d(\mu_{\psi^{(R)}}, \mu_\psi) \leq K'\gamma^R. \tag{4.34}$$

Thus, the Kullback-Leibler divergence decays exponentially fast with decay rate γ.
In practice, we can conclude that if we take a range of potential as

$$R \sim \frac{\log K'}{\log \gamma} \tag{4.35}$$

the approximation for a finite time memory of $R-1$ should be a valide one.

4.4.5 Kullback-Leibler divergence for BMS model

With the finite range potential approximation we are now able to find a probability measure for the BMS model, that we call μ_{BMS} for simplicity. Considering that the goal of our work is to extrapolate the statistics model behind empirical experimental rasters, μ_{BMS} should be able to approximate as best possible the hidden probability. As in section 3.1.4, we use the empirical average of the raster in place of μ and the Kullback-Leibler divergence as a criterion for estimate the "distance" between μ_{BMS} and $\pi_\omega^{(T)}(\omega_0^D)$ where $D = R-1$ is the memory depth. We have

$$d_{KL}(\mu_{\text{BMS}}, \pi_\omega^{(T)}) = P(\psi) - \pi_\omega^{(T)}(\psi) - h(\pi_\omega^{(T)}). \tag{4.36}$$

Recalling the purpose to approximate as the best the statistics of raster plots using the BMS model and to set at the same time the synaptic weights and the input current, we need to have the "distance" between the empirical measure of the raster and the BMS measure as near as possible. In the next section, we will explore the minimization of the KL divergence with respect to the synaptic weights and with respect to the input current, in order to approximate the hidden probability of rasters setting the two most important parameters of the model.

4.5 Minimizing Kullback-Leibler divergence

Most of the work in Neuromathcomp Department in INRIA, with the fundamental support of Bruno Cessac, was focused on the formalization and development of a method for fitting parameters model (synaptic weights and input current) based on the minimization of the Kullback-Leibler divergence between approximated and real distributions. Taking into account that most of examples for determining the synaptic weights in literature [37] has the following form

$$\delta W_{ij}(t+1) = \epsilon\, g\left(W_{ij}(t), \omega(t)\right) \tag{4.37}$$

where g is a suitable deterministic function, we develop the following dynamics where g takes the form of the gradient of Kullback-Leibler divergence.

4.5.1 Minimizing with respect to synaptic weights

As explained in section 2.3, a *plasticity rule* is a dynamics which corresponds to the evolution of the synaptic weights. In our notations W_{ij} provides the maximal amplitude of the post-synaptic potential induced, at the synapse connecting j to i, when neuron j spikes. Here we consider the following coupled dynamics. Neurons are evolving according to (4.6) and we focus on slow synapse dynamics of the form:

$$\begin{aligned} \delta \boldsymbol{W}^{(\tau)} &= \epsilon\, g\left(\boldsymbol{W}^{(\tau)}, \omega(t)\right) \\ \delta \boldsymbol{W}^{(\tau)} &= \boldsymbol{W}^{(\tau+1)} - \boldsymbol{W}^{(\tau)} \end{aligned} \tag{4.38}$$

where $\boldsymbol{W}^{(\tau)}$ is the synaptic weight matrix at plasticity step τ, ϵ is a general small parameter which maintains the dynamics slow respect to membrane potential dynamics and g is a function of the rasters produced by the network that in our case it will be defined thanks the minimization of KL divergence. Examples of plasticity rules are STDP and Hebbian Rule, section 2.4, and in litterature there are different approach to define this function, see [2] [18] [35].

We want to minimize (4.36) with respect to synaptic weights and we construct dynamics of the form

$$g\left(\boldsymbol{W}^{(\tau)}, \omega(t)\right) = -\nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} d_{KL}(\mu_{\text{BMS}}, \pi_{\omega}^{(T)}). \tag{4.39}$$

Thus, the temporal change of synaptic weights reads:

$$\frac{d\boldsymbol{W}^{(\tau)}}{dt} = -\epsilon \, \nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} \left(P(\psi) - \pi_{\omega}^{(T)}(\psi) - h(\pi_{\omega}^{(T)}) \right). \tag{4.40}$$

The entropy does not influence the gradient because it does not depend on weights. Concerning the normalized potential $P(\psi)$ we have:

$$\begin{aligned} P(\psi) &= 0, \\ \text{but} \quad \nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} P(\psi) &\neq 0, \end{aligned} \tag{4.41}$$

we can only say that

$$\nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} P(\psi) d\boldsymbol{W} = 0 \tag{4.42}$$

Then, (4.40) becomes

$$\frac{d\boldsymbol{W}^{(\tau)}}{dt} = \epsilon \left(\nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} \pi_{\omega}^{(T)}(\psi) - \nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}} P(\psi) \right). \tag{4.43}$$

Approximating ψ with the finite range potential, as in section 4.4.3, with memory depth D, we can write

$$\pi_{\omega}^{(T)}(\psi) = \sum_{\omega_0^D} \pi_{\omega}^{(T)}(\omega_0^D)\psi(\omega_0^D) \tag{4.44}$$

and then,

$$\begin{aligned} \nabla \pi_{\omega}^{(T)}(\psi) &= \sum_{\omega_0^D} \pi_{\omega}^{(T)}(\omega_0^D) \nabla \psi(\omega_0^D) \\ &= \pi_{\omega}^{(T)}(\nabla \psi), \end{aligned} \tag{4.45}$$

where we use $\nabla = \nabla_{\boldsymbol{W}=\boldsymbol{W}^{(\tau)}}$ in order to simplify the notation and where $\pi_{\omega}^{(T)}(\nabla\psi)$ is the empirical average of the gradient potential. From [27] we know that: if ψ and ϕ are real Lipschitz functions and μ is the equilibrium state of ψ, for all small perturbations δ, we have

$$\frac{d}{d\delta} P\left(\psi + \delta\phi\right)\Big|_{\delta=0} = \mu(\psi). \tag{4.46}$$

Thus, in our particular case we obtain

$$\begin{aligned} P\left(\psi(\boldsymbol{W}+d\boldsymbol{W})\right) &= P\left(\psi(\boldsymbol{W})+\nabla\psi d\boldsymbol{W}+o(d\boldsymbol{W}^2)\right) \\ &= P\left(\psi(\boldsymbol{W})\right)+\mu(\nabla\psi)d\boldsymbol{W}+o(d\boldsymbol{W}^2) \end{aligned} \tag{4.47}$$

Finally we can rewrite (4.43) as

$$\frac{d\boldsymbol{W}^{(\tau)}}{dt} = \epsilon\left(\nabla\pi_{\omega}^{(T)}(\psi)-\mu\left(\nabla P(\psi)\right)\right). \tag{4.48}$$

For the BMS model the potential with $D = R-1$ memory depth is

$$\begin{aligned} &\psi(\omega_0^D) = \\ &= \sum_{k=1}^{N}\left[\omega_k(D)\log\left[\Pi\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)\right]+(1-\omega_k(D))\log\left[1-\Pi\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)\right]\right], \end{aligned} \tag{4.49}$$

where $C_k(\omega_0^{D-1})$ depends on the synaptic weights as follows:

$$\begin{aligned} C_k(\omega_0^{D-1}) &= \sum_{h=1}^{N} W_{kh}x_{kh}(\omega_0^{D-1}) + I_k\frac{1-\gamma^{D-\tau_k(\omega_0^{D-1})}}{1-\gamma} \\ x_{kh}(\omega_0^{D-1}) &= \sum_{l=\tau_k(\omega_0^{D-1})}^{D-1}\gamma^{D-1-l}\omega_h(l). \end{aligned} \tag{4.50}$$

Then,

$$\begin{aligned} &\nabla\psi(\omega_0^D) = \\ &= \sum_{k=1}^{N}\left(\omega_k(D)\frac{\Pi'\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)}{\Pi\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)}-(1-\omega_k(D))\frac{\Pi'\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)}{\left[1-\Pi\left(\frac{\theta-C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)\right]}\right)\nabla C_k(\omega_0^{D-1}). \end{aligned} \tag{4.51}$$

Since

$$\begin{aligned} \left[\nabla C_k(\omega_0^{D-1})\right]_{ij} &= \frac{\partial}{\partial W_{ij}}\sum_{h=1}^{N} W_{kh}x_{kh}(\omega_0^{D-1}) = \\ &= x_{kh}(\omega_0^{D-1})\ \delta_{ik}\ \delta_{jh}, \end{aligned} \tag{4.52}$$

where $\delta_{ik} = 1$ if $i = k$, $\delta_{ik} = 0$ otherwise.
We obtain:

$$\frac{\partial}{\partial W_{ij}}\psi(\omega_0^D) = \\ = \sum_{k=1}^{N} \left((\omega_k(D) \frac{\Pi'\left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)}{\Pi\left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)} - (1 - \omega_k(D)) \frac{\Pi'\left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)}{\left[1 - \Pi\left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\right)\right]} \right) \cdot \\ \cdot \left[-\frac{x_{kh}(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})}\ \delta_{ik}\ \delta_{jh} \right] \tag{4.53}$$

Furthermore, we can approximate $\mu\left(\nabla P(\psi)\right)$ as in section 3.1.4 with the empirical average observed in the model: for $T \longrightarrow +\infty$ we have $\pi_{\text{BMS}}(\nabla\psi) \sim \mu(\nabla\psi)$. The difference between $\pi_{\text{BMS}}(\nabla\psi)$ and $\pi_{\omega}^{(T)}\left(\nabla\psi\right)$ will become clearer in next section, when we will deal with numerical simulations.
Finally, the dynamical equation reads:

$$\frac{dW_{ij}^{(\tau)}}{dt} = \epsilon \left(\pi_{\omega}^{(T)} \left(\frac{\partial}{\partial W_{ij}}\psi \right) - \pi_{\text{BMS}} \left(\frac{\partial}{\partial W_{ij}}\psi \right) \right). \tag{4.54}$$

4.5.2 With respect to input current

Using the same idea as in the previous section, we minimize the KL divergence with respect to the input current in order to set this parameter. Taking into account that now the temporal change in input current is:

$$\frac{dI}{dt} = -\epsilon\ \nabla_I \left(P(\psi) - \pi_{\omega}^{(T)}(\psi) - h(\pi_{\omega}^{(T)}) \right), \tag{4.55}$$

and the derivative with respect to the current of the potential is

$$\begin{aligned}
&\frac{\partial}{\partial I_i}\psi(\omega_0^D) = \\
&= \sum_{k=1}^{N} \left((\omega_k(D) \frac{\Pi' \left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})} \right)}{\Pi \left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})} \right)} - (1 - \omega_k(D)) \frac{\Pi' \left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})} \right)}{\left[1 - \Pi \left(\frac{\theta - C_k(\omega_0^{D-1})}{\sigma_k(\omega_0^{D-1})} \right) \right]} \right) \cdot \\
&\cdot \left[-\frac{1 - \gamma^{D - \tau_k(\omega_0^{D-1})}}{(1-\gamma)\sigma_k(\omega_0^{D-1})} \, \delta_{ik} \right]
\end{aligned} \tag{4.56}$$

the change in input current reads:

$$\frac{dI_i}{dt} = \epsilon \left(\pi_\omega^{(T)} \left(\frac{\partial}{\partial I_i} \psi \right) - \pi_{\text{BMS}} \left(\frac{\partial}{\partial I_i} \psi \right) \right). \tag{4.57}$$

4.5.3 Remarks

We provide a method to set synaptic weights and input current that requires the KL divergence between real and approximated probabilities to converge to zero. This convergence will be verified in section 6.2 after having discussed the numerical implementation in C++ of equations (4.54) and (4.57) and other implementation details which are important to understand the numerical context (BMS model and its parameters) behind this method. We will extrapolate the synaptic weights during the time steps of simulations as follows:

$$\begin{aligned}
W_{ij}^{(\tau+1)} &= W_{ij}^{(\tau)} + \frac{dW_{ij}^{(\tau)}}{dt} = W_{ij}^{(\tau)} + \epsilon \left(\pi_\omega^{(T)} \left(\frac{\partial}{\partial W_{ij}} \psi \right) - \pi_{\text{BMS}} \left(\frac{\partial}{\partial W_{ij}} \psi \right) \right), \\
I_{ij}^{(\tau+1)} &= I_{ij}^{(\tau)} + \frac{dI_{ij}^{(\tau)}}{dt} = I_{ij}^{(\tau)} + \epsilon \left(\pi_\omega^{(T)} \left(\frac{\partial}{\partial I_i} \psi \right) - \pi_{\text{BMS}} \left(\frac{\partial}{\partial I_i} \psi \right) \right).
\end{aligned} \tag{4.58}$$

Chapter 5

Numerical implementation

5.1 BMS model

The BMS model was implemented in C++ by the Neuromathcomp team in INRIA. First of all they created a class of BMS membrane potential composed according to (4.6) that can simulate a neuron membrane potential and spike trains.

5.1.1 Members of BMS membrane potential class

In this class the public parameters are:

- *number_of_units* is the number of neurons N;
- *unit_leak* is the decay rate or "*leak rate*" γ;
- *unit_currents* is the external input current applied to neuron i, $\mathbf{I} = (I_i)_{i=1}^N$;
- *unit_noise* is the additive noise $\mathbf{B(t)} = (B_i(t))_{i=1}^N$, distributed like a Gaussian with zero mean and variance σ_B;
- W is the matrix of the synaptic weights W_{ij}.

5.1.2 Methods of BMS membrane potential class

This class is composed of functions that are reported here for sake of completeness.

Reset function

```
BMSPotential& BMSPotential::reset(unsigned int N, double
    leak, double sigmaB, const double *Ie)
  {
    sys::check(0 <= leak && leak < 1, "in
       BMSPotential::reset the leak term is not in the
       interval [0,1[ -> %f", leak);
    sys::check(0 < sigmaB, "in BMSPotential::reset the
       noise standard deviation sigmaB is negative -> %f",
       sigmaB);
    number_of_units = N;
    int R=10000; //Infinite range
    resetEmpty(N,R);
    unit_leak = leak;
    delete[] unit_currents;
    unit_currents = new double[number_of_units];

    for(unsigned int i = 0; i < number_of_units; i++)
         unit_currents[i] = Ie == NULL ? 0 : Ie[i];

    unit_noise = sigmaB;
    delete[] W;
    W = new double[N * N];

    for (int oi=0;oi<N*N;oi++)
         W[oi]=0.0;

    return *this;
  }
```

This function resets the parameters of the membrane potential $\boldsymbol{V} = (V_i)_{i=1}^N$ according to (4.6), given in input the number of neurons N, the leak γ, the noise amplitude σ_B and the current I. Moreover this function sets all the synaptic weights to zero.

Set up of synaptic weights

There are different kinds of functions for taking in input random weights depending on what the user needs.

```
void BMSPotential::setWeight(unsigned int i, unsigned int
    j, double value)
  {
    sys::check(i < number_of_units, "in
       RNLunits::setWeight, bad post-synaptic neuron index
       %d not in {0, %d{", i, number_of_units);
    sys::check(j < number_of_units, "in
       RNLunits::setWeight, bad  pre-synaptic neuron index
       %d not in {0, %d{", j, number_of_units);
    W[i * number_of_units + j] = value;
  }
```

This function sets the synaptic weight from neuron j to neuron i to a given value decided by the user.

```
BMSPotential& BMSPotential::setWeights(double sigma, double
    bias)
  {
    double mean = sys::getGaussianBias(bias, sigma);
    N=number_of_units;

    for(unsigned int oi=0; oi<N*N; oi++)
          W[oi] = sys::gaussian(mean, sigma);


   return *this;
  }
```

This function sets Gaussian random value to the synaptic weights according to the bias and the variance given in input. The bias is a value in $(0, 1)$ that points out how many weights have to be negative or positive, *e.g.* 0.5 of bias means that 50% of weights will be positive and 50% will be negative. All the values of synaptic weights will fluctuate in the interval $[-sigma,\ sigma]$.

```
BMSPotential& BMSPotential::setSparseWeights(int K, double
    sigma, double bias)
  {
    double mean = sys::getGaussianBias(bias, sigma);
    int vois[K];
    for(unsigned int oi=0; oi<number_of_units; oi++){
         int nbvois=0;
         while (nbvois<K){
              uint idx=sys::random(0,number_of_units);
              int j=0;
              while ((j<nbvois)&&(idx!=vois[j]))
                   j++;
              if (j==nbvois){
                   vois[nbvois]=idx;
                   nbvois++;
              }
         }
        for (int j=0;j<K;j++){
             W[oi*number_of_units+vois[j]] =
              =sys::gaussian(mean, sigma);
        }
    }
    return *this;
 }
```

This function does the same as the previous one with the difference that only K neurons for all $i = 1, \ldots, N$ are selected randomly and set with a Gaussian weight.

Finally, the following function returns the weight corresponding to the connection from j to i:

```
  double BMSPotential::getWeight(unsigned int i, unsigned
     int j) const
{
    sys::check(i < number_of_units, "in RNLunits::getWeight,
                 bad post-synaptic neuron index %d not in
                 {0, %d{", i, number_of_units);
    sys::check(j < number_of_units, "in RNLunits::getWeight,
                 bad pre-synaptic neuron index %d not in
                {0, %d{", j, number_of_units);
    return W[i * number_of_units + j];
  }
```

Generate a Raster based on BMS model

```
  RasterBlock *BMSPotential::getRasterBlock(unsigned int
     transients, unsigned int length) const
  {
    RasterBlock *raster = new RasterBlock();
    raster->reset(number_of_units, length);
 //Initial potential
    double *Vvp = new double[number_of_units];
    double *Vv = new double[number_of_units];

    for(unsigned int i=0; i<number_of_units; i++)
        Vv[i] = 2*sys::random();
 //Iterates on time and units: transients
    for(unsigned int t=0; t<transients; t++) {
         for(unsigned int i=0; i<number_of_units; i++) {
              Vvp[i]=unit_currents[i]+unit_noise *
                        *sys::gaussian();
              for(unsigned int j=0; j<number_of_units; j++)
                    if (Vv[j] >= unit_threshold)
                          Vvp[i] += getWeight(i,j);
                    if(Vv[i] < unit_threshold)
                          Vvp[i] += unit_leak * Vv[i];
       }
     //Pingpong the buffers
        double *V = Vv;
        Vv = Vvp;
        Vvp = V;
```

```
  }
//Iterates on time and units: stores the raster
 for(unsigned int t = 0; t < length; t++) {
      for(unsigned int i = 0; i < number_of_units; i++) {
          Vvp[i]=unit_currents[i]+unit_noise*
                      *sys::gaussian();
          for(unsigned int j=0; j<number_of_units; j++)
               if (Vv[j]>= unit_threshold)
                   Vvp[i]+=getWeight(i,j);
               if(Vv[i] >= unit_threshold)
                    aster->setEvent(i, t, true);
               else
                    Vvp[i] += unit_leak * Vv[i];
       }
    //Pingpong the buffers
       double *V = Vv;
       Vv = Vvp;
       Vvp = V;
   }
  delete[] Vv;
  delete[] Vvp;
  return raster;
}
```

First of all it is necessary to underline that there exists a class of raster developed by INRIA's team called *RasterBlock*. This class generates raster plots according to section 3.1.1.

Thus, the function above returns a raster plot of spikes according to BMS model dynamics. The mechanism is as follows. An event (a spike) is set every time the membrane potential of neuron i exceeds the threshold. This threshold is overtaken thanks to contributions of currents, noise and other neurons. Namely the membrane potential of neuron i is the sum of contributions coming from the current (set at the beginning), the noise and the respective weights of all neurons j, $j = 1, \ldots, N$, whenever these neurons exceed in turn the threshold. On the contrary, if neuron j does not fire, the dynamics are only influenced by the decay rate γ, as we could easily infer observing equation (4.6).

This procedure is repeated twice. Once for a time length *transient* and once for *length* time, with the fundamental difference that the second step does not begin with random values for membrane potentials, but with the result obtained during the *transient* time.

Conditional probability

```
double
   BMSPotential::getEventConditionalProbability(unsigned
   int k, bool omegak,const RasterBlock& word) const
  {
    sys::check(word.getNumberOfUnits() == number_of_units,
         "in BMSPotential::getEventConditionalProbability
         uncoherent block size %d != %d the units number",
         word.getNumberOfUnits(), number_of_units);
    sys::check(word.getNumberOfTimeSteps() > 0, "in
         BMSPotential::getEventConditionalProbability
         word range = %d must be >= ",
         word.getNumberOfTimeSteps());

    if (omegak==1)
        return pi(getX(k,word));
    else
        return (1-pi(getX(k,word)));
  }
```

This function returns the probability that a neuron k fires or does not fire at time D, depending on the value of *omega* given in input (1 or 0 respectively), subject to the previous events plot in the raster *word*. It is implemented according to (4.17), where *getX* is $\frac{\theta - C_i(\omega_0^{D-1})}{\sigma_i}$ and *pi* is $\frac{1}{\sqrt{2\Pi}} \int_x^{+\infty} e^{\frac{-u^2}{2}} du$ that are implemented as follows:

```
double BMSPotential::getX(unsigned int i, const
   RasterBlock& word) const {
    double X=0;
    double sigma=getSigma(i,word);
    sys::check(sigma>0, "in BMSPotential::getX : the
        cumulative noise variance = %lg must be >0", sigma);

    for (unsigned int j=0;j<number_of_units;j++)
       X+= getWeight(i,j)*getEta(i,j,word);

    double Y=(unit_threshold-X-getIe(i,word))/sigma;

    return Y;
  }
```

and

```
double BMSPotential::pi(double x) const
  {
    return 0.5 * (1 - Erf(x / sqrt(2.0)));
  }
```

where *getSigma* extrapolates $\sigma_i(\omega_0^{D-1}) = \sqrt{\sigma_B^2 \frac{1-\gamma^{2(D-\tau_i(\omega_0^{D-1}))}}{1-\gamma^2}}$ in (4.17)

```
double BMSPotential::getSigma(unsigned int i, const
    RasterBlock& word) const {
     int D=word.getNumberOfTimeSteps();
     double sigma=unit_noise*sqrt( (1-pow(unit_leak,2*
       *(D-getTau(i,word))))/(1-pow(unit_leak,2)) );
     return  sigma;
  }
```

getTau finds the last firing rate $\tau_i(\omega_0^{D-1})$ in (4.7)

```
unsigned int BMSPotential::getTau(unsigned int i, const
    RasterBlock& word) const {
     int D=word.getNumberOfTimeSteps();
     sys::check(D >= 1, "in BMSPotential::getTau :
           range of the condition = %d must be >= 1", D);

     for (int t=D-1;t>=0;t--)
          if (word.getEvent(i,t)) return t;
     return 0;
  }
```

getIe implements the current $I_i \frac{1-\gamma^{D-\tau_i(\omega_0^{D-1})}}{1-\gamma}$

```
double BMSPotential::getIe(unsigned int i, const
    RasterBlock& word) const {
     int D=word.getNumberOfTimeSteps();
     double Ie=unit_currents[i]*
       *(1-pow(unit_leak,D-getTau(i,word)))/(1-unit_leak);
     return  Ie;
  }
```

and finally, *getEta* calculates $\xi_i(\omega_0^{D-1}) = \sum_{l=\tau_i(\omega_0^{D-1})}^{D-1} \gamma^{D-1-l} B_i(l)$

```
double BMSPotential::getEta(unsigned int i,unsigned int j,
    const RasterBlock& word) const {
     double eta=0;
     int D=word.getNumberOfTimeSteps();
     sys::check("D>=1","in BMSPotential::getEta.
            The word range %d must be larger than 1",D);
     unsigned int Dmoins1=D-1;

     for (unsigned int l=getTau(i,word);l<=Dmoins1;l++)
         if (word.getEvent(j,l))
             eta+=pow(unit_leak,Dmoins1-l);
     return eta;
   }
```

Then, for an entire spiking pattern the condition probability is implemented as follows:

```
double BMSPotential::getConditionalProbability(const
    RasterBlock& event, const RasterBlock& condition) const
  {
     sys::check(condition.getNumberOfUnits() ==
        number_of_units,
       "in BMSPotential::getConditionalProbability uncoherent
        block size %d != %d the units number",
        condition.getNumberOfUnits(), number_of_units);
     sys::check(condition.getNumberOfTimeSteps() > 1,
        "in BMSPotential::getConditionalProbability word
           range =
        %d must be >= 2", condition.getNumberOfTimeSteps());
     double P=1;
     for (unsigned int k=0;k<number_of_units;k++)
          if (event.getEvent(k,0)==1)
              P*=pi(getX(k,condition));
          else
              P*=(1-pi(getX(k,condition)));
     return P;
   }
```

This function takes into account the approximation of conditional independence between the $V_i(D-1)$'s as proved in section 4.3.2. Moreover, it requires in input two different rasters: the *event* and the *conditon*. The event

is a raster of one time step (at time D) of which we would like to measure the conditional probability, namely it tells us if $\omega_i(D)$ is one or zero for all $i = 1, \ldots, N$. The condition, as the name suggests, is the raster that gives parameters in order to reach the final probability, *i.e.* where σ_i, C_i and pi are extrapolated.

In BMS potential class there is another function that calculates the conditional probability of a spiking pattern without asking in input the condition and the event raster and it reads as follows:

```
double BMSPotential::getConditionalProbability(const
   RasterBlock& w)
  {
    uint R=w.getNumberOfTimeSteps(),D=R-1;
    sys::check(w.getNumberOfUnits() == number_of_units,
           "in BMSPotential::getEventConditionalProbability
           uncoherent block size %d != %d the units number",
           w.getNumberOfUnits(), number_of_units);
    sys::check(R > 0, "in
       BMSPotential::getEventConditionalProbability
           word range = %d must be >= 0", R);
    double P=1;
    RasterBlock event;
    event.resetSubSequence(w,D,1);
    RasterBlock condition;
    condition.resetSubSequence(w,0,D);
    for (unsigned int k=0;k<number_of_units;k++)
         if (event.getEvent(k,0)==1)
             P*=pi(getX(k,condition));
        else
             P*=(1-pi(getX(k,condition)));
    return P;
  }
```

The raster where it is checked the condition and the raster of events are directly generated by the function, which creates two new rasters from the initial one with length time respectively of D and 1.

Potential implementation

```
double BMSPotential::Phi(const RasterBlock& w) const
  {
    uint R=w.getNumberOfTimeSteps(),D=R-1;
    sys::check(w.getNumberOfUnits() == number_of_units,
      "in BMSPotential::Phi uncoherent block size %d != %d
      the units number", w.getNumberOfUnits(),
          number_of_units);
    sys::check(R > 0, "in BMSPotential::Phi word range = %d
       must be >= 0", R);
    RasterBlock event;
    event.resetSubSequence(w,D,1);
    RasterBlock condition;
    condition.resetSubSequence(w,0,D);
    double fi=0;
    for (unsigned int k=0;k<number_of_units;k++) {
        if (event.getEvent(k,0)==1)
           fi+=lpi(getX(k,condition));
        else
           fi+=ilpi(getX(k,condition));
    }
    return fi;
  }
```

This function implements the potential according to equation (4.29), using the same method of the previous function for the evaluation of the conditional probability. We have that *lpi* and *ilpi* read:

```
double BMSPotential::lpi(double x) const
  {
    if (x<-tol_pi) //Taylor expansion near -\infty
      return exp(-pow(x,2)/2)*(3989422802/x-
         -.3989422802/pow(x,3)+1.196826841/pow(x,5)-
         -.984134204/pow(x,7)+41.88893942/pow(x,9)-
         -377.0004548/pow(x,11)+4147.005003/pow(x,13));

    if (x>tol_pi) //Taylor expansion near +\infty
      return -0.5*pow(x,2)-0.9189385335-log(x)-1/pow(x,2)+
       +2.5/pow(x,4)-12.33333333/pow(x,6)+88.25/pow(x,8)-
        -816.2000000/pow(x,10)+9200.833335/pow(x,12);

   return log(pi(x));
  }
```

This is the function for *lpi* is equal to $\log\left(\Pi\left[\frac{\theta-C_i(\omega_0^{D-1})}{\sigma_i(\omega_0^{D-1})}\right]\right)$ in (4.29) and

```
double BMSPotential::ilpi(double x) const
  {
    if (x<-tol_pi)
        return -
         0.5*pow(x,2)-0.9189385335-log(-x)-1/pow(x,2)+
          +2.5/pow(x,4)-12.33333333/pow(x,6)+88.25/pow(x,8)-
          -816.2000000/pow(x,10)+9200.833335/pow(x,12);

    if (x>tol_pi)
        return exp(-pow(x,2)/2)*(-.3989422802/x+
           +.3989422802/pow(x,3)-1.196826841/pow(x,5)+
           +5.984134204/pow(x,7)-41.88893942/pow(x,9)+
           +377.0004548/pow(x,11)-4147.005003/pow(x,13));

    return log(1-pi(x));
  }
```

ilpi is equal to $\log\left(1-\Pi\left[\frac{\theta-C_i(\omega_0^{D-1})}{\sigma_i(\omega_0^{D-1})}\right]\right)$ in (4.29).

Once we have implemented the potential (4.29), we need to implement the Kullback-Leibler divergence from (4.36) taking into account that the topological pressure (3.18) for a normalized potential is equal to zero.

Kullback-Leibler divergence implementation

Below we have the function that implements the Kullback-Leibler divergence between the probability distribution of rasters in BMS model and the empirical probability distribution.

```
double BMSPotential::getDivergence(RasterBlock *Raster,uint
    R)
  {
    double dist=0;
    RasterBlockGrammar grammar;
    grammar.reset(*Raster, R);

    for(RasterBlockIterator& i = grammar.reset();
        i.hasNext();){
          const RasterBlock& word = i.getItem();
          dist-=Phi(word)*grammar.getJoinProbability(word);
    }

   return dist-grammar.getStrongEtAlEntropy();
  }
```

To better understand this function it is necessary to explain what is a *grammar* and the idea behind it.

Grammar

The dynamical system (4.6) (with noise) can in principle produce any pattern sequence (although with different probability). For N neurons and spike blocks of size R, this means 2^{NR} possibilities, which is quite huge. On the opposite, considering a raster of size $T \sim 10^6$, dynamics will produce, at most, $T - R$ spike blocks of size R. Thus, the idea is to store only the observed blocks and their transitions in order to estimate the transition probabilities. Let us explain it clearer with an example. Suppose to have a raster plot of this form

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & \dots \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & \dots \end{bmatrix} \tag{5.1}$$

with D=1 (R=2), so all $\omega(1)$ should be $2^{2\cdot 2}$ possibilities, it seems not too much but if we have hundreds of neurons and a bigger range R, we should

have 2^{100R} and this is huge! So the idea is to consider only the *legal* or *admissible* transitions that are recorded from the raster plot. In our simple case the possibilities will be only: if a step forward we have the prefix $\begin{bmatrix}\mathbf{0}\\\mathbf{0}\end{bmatrix}$, all the possible transitions are $\begin{bmatrix}1\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\end{bmatrix}$ and $\begin{bmatrix}1\\1\end{bmatrix}$; if our prefix is $\begin{bmatrix}\mathbf{1}\\\mathbf{0}\end{bmatrix}$, the only possible transition is $\begin{bmatrix}1\\1\end{bmatrix}$; if a step forward we have $\begin{bmatrix}\mathbf{1}\\\mathbf{1}\end{bmatrix}$, the admissible transition is $\begin{bmatrix}0\\0\end{bmatrix}$; and finally if a step forward the prefix is $\begin{bmatrix}\mathbf{0}\\\mathbf{1}\end{bmatrix}$ the transition is $\begin{bmatrix}0\\0\end{bmatrix}$. In this way we reduce the number of transition that are observed. That means we have only 6 different possibilities instead of 16.
Let now formalize what is stated above.
Let ω, ω' be two spiking patterns, the transition $\omega \to \omega'$ is *legal* or *admissible* if there exists a neural state $\mathbf{X}$ such that neurons fire according to the firing pattern ω and, the next time, according to the firing pattern ω'. If there exists a finite integer r such that it is possible to construct a block of spiking patterns $\omega(0)\dots\omega(r-1)$, there are at most 2^{Nr} possible blocks. Label each of these with a symbol $\alpha \in A$, where A is called an *alphabet*. Define a transition matrix $G_\gamma : A \times A \to \{0,1\}$, depending on the leak γ, with entries $g_{\alpha\beta}$, such that $g_{\alpha\beta}$=1 if the transition $\alpha \to \beta$ is admissible, and 0 otherwise. To alleviate notations in the next equation write $\alpha(t) = [\omega]_t^{t+r-1}, \forall t \geq 0$. If

$$\Sigma_\gamma = \{\omega | g_{\alpha(t+1)\alpha(t)} = 1, \forall t \geq 0\}, \tag{5.2}$$

then all admissible raster plots are obtained via the transition matrix G_γ, which provides the grammar of the spiking pattern sequences.

In practice, this kind of "mechanism" is used to calculate the empirical average of the raster and it improves considerably the implementation. Thus, in Kullback-Leibler divergence function it is created the grammar tree from the raster given in input and, finally, the equation (4.36) is implemented.

5.2 Kullback-Leibler divergence minimization

During my internship in INRIA we developed in C++ the method to extrapolate weights from minimization of Kullback-Leibler divergence according to section 4.5. For implementing the equation (4.53) we have to evaluate first $\Pi'\left(\frac{\theta-C_i(\omega_0^{D-1})}{\sigma_i(\omega_0^{D-1})}\right)$. For sake of simplicity, we define $\chi = \frac{\theta-C_i(\omega_0^{D-1})}{\sigma_i(\omega_0^{D-1})}$ and we have:

$$\Pi(\chi) = \frac{1}{\sqrt{2\pi}} \int_{\chi}^{+\infty} e^{-\frac{u^2}{2}} du \tag{5.3}$$

and its derivative

$$\Pi'(\chi) = -\frac{1}{\sqrt{2\pi}} e^{-\frac{\chi^2}{2}}. \tag{5.4}$$

Then, from (4.53) we find:

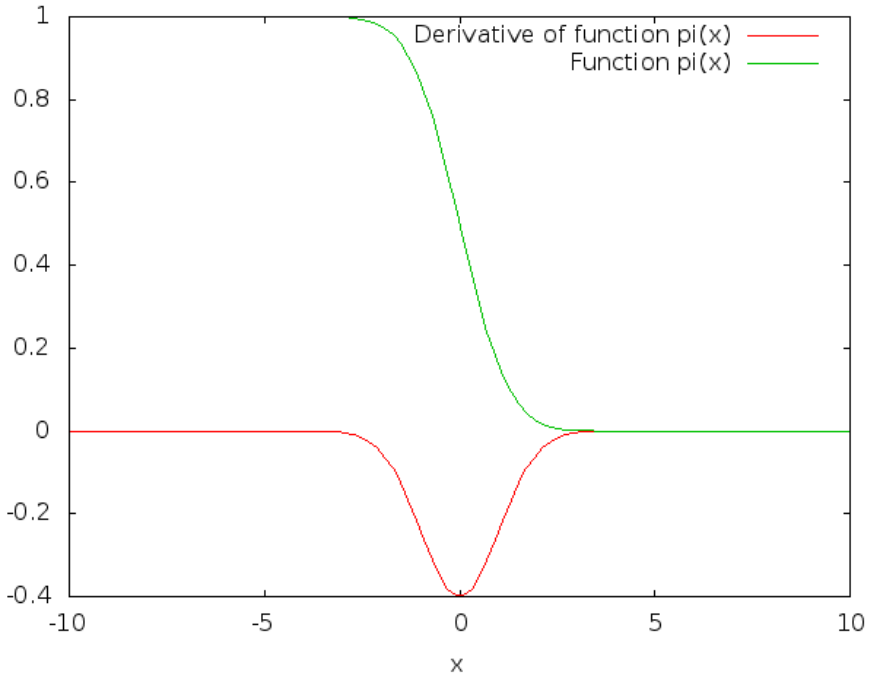
$$\begin{aligned} \frac{\Pi'(\chi)}{\Pi(\chi)} &= \frac{-\frac{1}{\sqrt{2\pi}} e^{-\frac{\chi^2}{2}}}{\frac{1}{\sqrt{2\pi}} \int_{\chi}^{+\infty} e^{-\frac{u^2}{2}} du} \\ \frac{\Pi'(\chi)}{1-\Pi(\chi)} &= \frac{-\frac{1}{\sqrt{2\pi}} e^{-\frac{\chi^2}{2}}}{1-\frac{1}{\sqrt{2\pi}} \int_{\chi}^{+\infty} e^{-\frac{u^2}{2}} du}. \end{aligned} \tag{5.5}$$

Concerning the implementation of the two functions above, we used, as did before in section 5.1.2 the *Erf*, the error function.

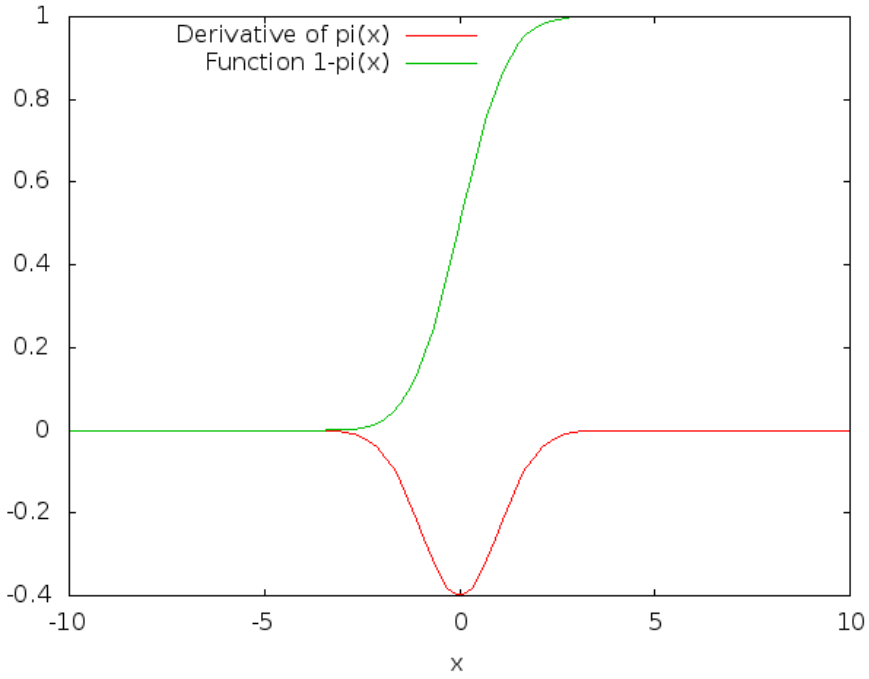
$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du, \tag{5.6}$$

so (5.3) becomes

$$\begin{aligned} \Pi(\chi) &= 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\chi} e^{-\frac{u^2}{2}} du = 1 - \frac{1}{\sqrt{2\pi}} \left[\int_{-\infty}^{0} e^{-\frac{u^2}{2}} du + \int_0^{\chi} e^{-\frac{u^2}{2}} du \right] = \\ &= 1 - \frac{1}{2} - \frac{1}{\sqrt{2\pi}} \int_0^{\chi} e^{-\frac{u^2}{2}} du = \frac{1}{2} \left[1 - \mathrm{Erf}\left(\frac{\chi}{\sqrt{2}} \right) \right] \end{aligned} \tag{5.7}$$



(a) $\Pi'(x)$ in red and $\Pi(x)$ in green.



(b) $\Pi'(x)$ in red and $1 - \Pi(x)$ in green.

Figure 5.1: Plot of the *error function* and its derivative

Thus, studying mathematically the ratio $\frac{\Pi'}{\Pi}$ we note that when $\chi \to +\infty$ both Π and its derivative Π' go to zero as show in figure 5.1. Even if it is easy to calculate the limit analytically, for example with l'Hospital's rule as follows

$$\lim_{\chi\to+\infty} \frac{-\frac{1}{\sqrt{2\pi}}e^{-\frac{\chi^2}{2}}}{\frac{1}{\sqrt{2\pi}}\int_{\chi}^{+\infty} e^{-\frac{u^2}{2}}du} = \lim_{\chi\to+\infty} \frac{\chi e^{-\frac{\chi^2}{2}}}{-e^{-\frac{\chi^2}{2}}} = \lim_{\chi\to+\infty} -\chi = -\infty, \tag{5.8}$$

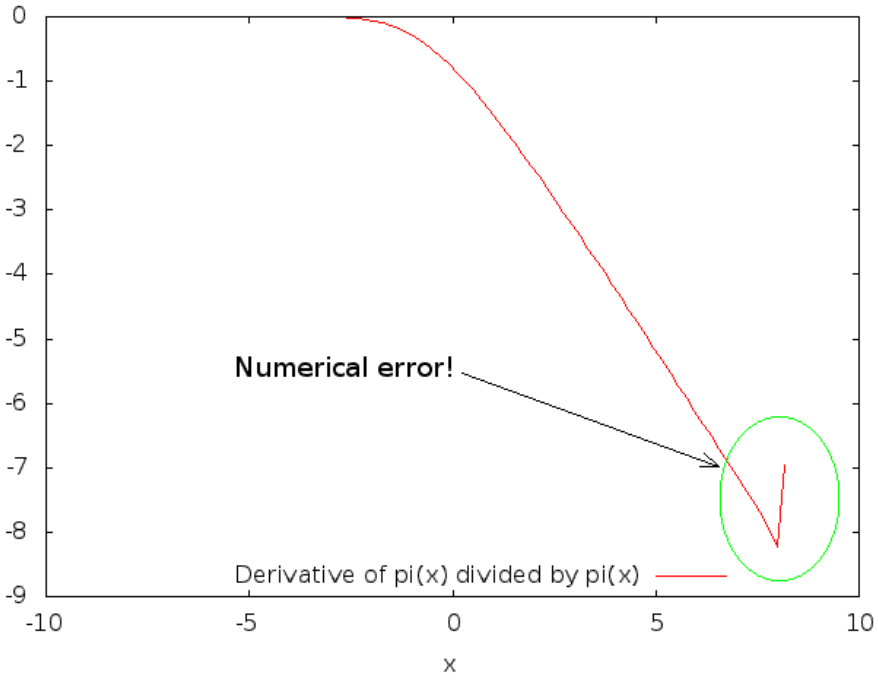
we can see from figure 5.2 that problems with numerical implementation start appearing for a very small values of χ, approximately in $[6, +\infty)$. Thus, it becomes necessary to replace the ratio in this interval with the asymptotic approximation. In [36] and [11] it is shown how to calculate the asymptotic expansion for the complementary of the Erf function ($\mathrm{Erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^{+\infty} e^{-u^2}du$). First it is possible to write it as a particulare case of *gamma* function $\mathrm{Erfc}(x) = \frac{2}{\sqrt{\pi}}\Gamma\left(\frac{1}{2}, x^2\right)$ and its asymptotic expansion is known:

$$\mathrm{Erfc}(x) \sim \frac{e^{-x^2}}{x\sqrt{\pi}}\left[1 - \frac{1}{2x^2} + \frac{1\cdot 3}{x^{2\cdot 2}} - \frac{1\cdot 3\cdot 5}{x^{2\cdot 3}} + \dots\right]. \tag{5.9}$$

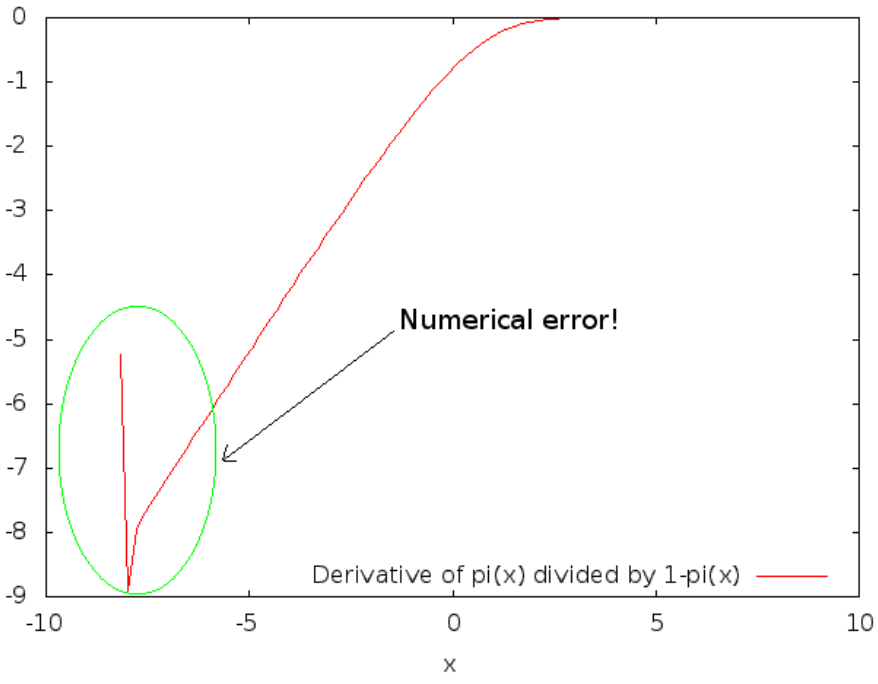
Finally, we can make the ratio between its derivative and the function above and we obtain the approximation to use in C++ code in the interval $[6, \infty)$.

Concerning the second relation in (5.5) we did the same for $\chi \longrightarrow -\infty$ and we compute the asymptotic approximation in the intervall $(-\infty, -6]$. Figure 5.3 shows the accuracy of the approximation with asymptotic expansions. The code in C++ for the implementation of $\frac{\Pi'}{\Pi}(x)$ and $\frac{\Pi'}{1-\Pi}(x)$ to add to the BMS potential class is below:

```
double BMSPotential::pi_prime_divby_pi(double x) const
{
 if(x>tol_pi){
   return (-0.9999999992*x-0.9999999992/x+
   +1.999999999/pow(x,3)-9.999999992/pow(x,5)+
   +73.99999996/pow(x,7)-705.9999997/pow(x,9)+
   +8161.999994/pow(x,11)-1.104099999*pow(10,5)/pow(x,13)+
   +1.708393999*pow(10,6)/pow(x,15));
 }
 else
    return  -one_over_sqrt2pi*exp(-pow(x,2)/2)/pi(x);
}
```



(a) $\frac{\Pi'(x)}{\Pi(x)}$



(b) $\frac{\Pi'(x)}{1-\Pi(x)}$

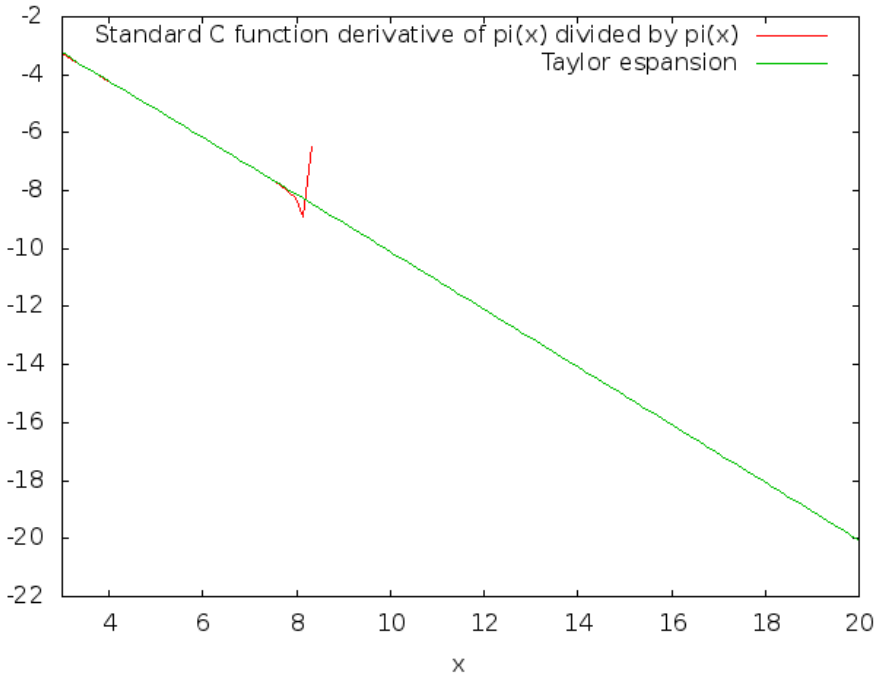
Figure 5.2: Plots that show numerical error when we calculate the ratio between error function and its derivative

where *tol_pi* is equal to 6 and

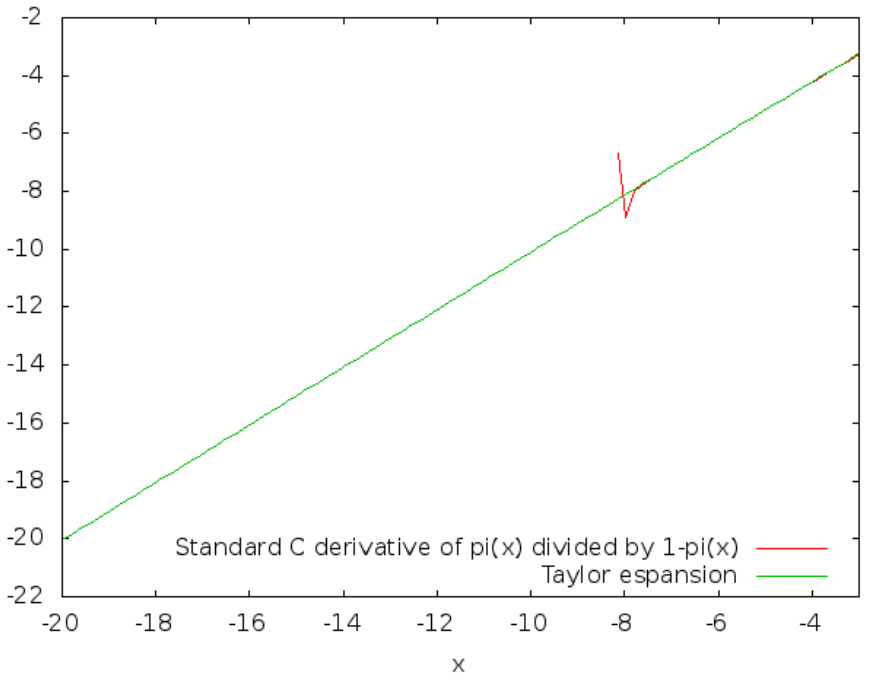
```
double BMSPotential::pi_prime_divby_one_minus_pi(double
    x) const
{
if(x<=-tol_pi){
  return (x+1/x-2.000000001/pow(x,3)+10.00000001/pow(x,5)-
   -73.99999996/pow(x,7)+706.0000002/pow(x,9)-
   -8162.000006/pow(x,11)+1.104100000*pow(10,5)/pow(x,13)-
   -1.708394001*pow(10,6)/pow(x,15));
}
else
   return  -one_over_sqrt2pi*exp(-pow(x,2)/2)/(1-pi(x));
}
```

Once we have solved the computational problems, we implement the gradient of the potential (4.53) as follows:

```
double *BMSPotential::getGradientDivergence
   (RasterBlockGrammar
   *grammarSought, RasterBlockGrammar*grammarBMS, unsigned
      int R)
{
    /* grammarSought is generated by the raster we'd like
       to approx */
    /* grammarBMS is the raster that change every time we
       iterate the method*/

   uint N=number_of_units;
   double *Gdkl=new double[N*N];
   for(unsigned i=0;i<N;i++)
       for(unsigned j=0;j<N;j++)
            Gdkl[i*N+j]=0;

 /** Computing gradient **/
   uint D=R-1;
   for(RasterBlockIterator& it = grammar->reset();
      it.hasNext();){
       const RasterBlock& word = it.getItem();

       RasterBlock event;
       event.resetSubSequence(word,D,1);

       RasterBlock condition;
```



(a) $\frac{\Pi'(x)}{\Pi(x)}$



(b) $\frac{\Pi'(x)}{1-\Pi(x)}$

Figure 5.3: Plot of functions in C++ and their respective asymptotic approximations.

```
        condition.resetSubSequence(word,0,D);

        double Pjoin=grammar->getJoinProbability(word);

             for(unsigned i=0;i<N;i++){
                 double ui=0,xi=getX(i,condition);
                 if (event.getEvent(i,0)==1) {
                    ui=-pi_prime_divby_pi(xi);
                  }
                  else {
                    ui=pi_prime_divby_one_minus_pi(xi);
                  }
                 ui=ui*Pjoin/getSigma(i,condition);
                 for(unsigned j=0;j<N;j++){
                     Gdkl[i*N+j]+=ui*getEta(i,j,condition);
                 }
             }
       }
    /** computing the gradient of the pressure **/
    for(RasterBlockIterator& k = grammarBMS->reset();
       k.hasNext();){
        const RasterBlock& wordBMS = k.getItem();

        RasterBlock eventBMS;
        eventBMS.resetSubSequence(wordBMS,D,1);

        RasterBlock conditionBMS;
        conditionBMS.resetSubSequence(wordBMS,0,D);

        double
           PjoinBMS=grammarBMS->getJoinProbability(wordBMS);

        for(unsigned i=0;i<N;i++){
             double uiBMS=0,xiBMS=getX(i,conditionBMS);
             if (eventBMS.getEvent(i,0)==1) {
                 uiBMS=-pi_prime_divby_pi(xiBMS);
             }
             else {
                 uiBMS=pi_prime_divby_one_minus_pi(xiBMS);
            }
           uiBMS=uiBMS*PjoinBMS/getSigma(i,conditionBMS);

           for(unsigned j=0;j<N;j++){
               Gdkl[i*N+j]-=uiBMS*getEta(i,j,conditionBMS);
```

```
            }
        }
    }
   /** Normalizing gradient **/
    double norm=0;
    for(unsigned int i = 0; i < N; i++){
        for(unsigned int j= 0; j < N; j++){
            norm+=pow(Gdkl[i*N+j],2);
         }
     }
    norm=sqrt(norm);
    if (norm > 1)
        for(unsigned int i = 0; i < N; i++){
            for(unsigned int j= 0; j < N; j++){
                Gdkl[i*N+j]/=norm;
            }
         }
    for(unsigned int i = 0; i < N; i++){
        for(unsigned int j= 0; j < N; j++){
            printf(" %5.3lg ", Gdkl[i*N+j]);
         }
        printf("\n");
     }
    return Gdkl;
 }
```

and finally the set up of synaptic weights reads:

```
BMSPotential& BMSPotential::setWeightsDivergence
   (RasterBlockGrammar* grammar, RasterBlockGrammar *
   grammarBMS, unsigned int R){

   double *G=getGradientDivergence(grammar,grammarBMS,R);
   uint N=number_of_units;

      for (uint i=0;i<N;i++)
             for (uint j=0;j<N;j++)
                 setWeight(i,j,W[i*N+j]+0.1*G[i*N+j]);

   delete []G;
}
```

Concerning the minimization of the KL divergence with respect to the input current and the set up of this, the procedure is basically the same as it was for synaptic weights. The code for the gradient of the potential (4.56) with respect to input current is below:

```
double *BMSPotential::getGradientDivergenceCurrent
   (RasterBlockGrammar *grammarSought, RasterBlockGrammar*
   grammarBMS,unsigned int R) {

    uint N=number_of_units;
    double *GIdkl=new double[N];
    for(unsigned i=0;i<N;i++) GIdkl[i]=0;

 /** Computing gradient **/
    uint D=R-1;
    for(RasterBlockIterator& it = grammarSought->reset();
       it.hasNext();){
        const RasterBlock& word = it.getItem();
        RasterBlock event;
        event.resetSubSequence(word,D,1);
        RasterBlock condition;
        condition.resetSubSequence(word,0,D);
        double
           Pjoin=grammarSought->getJoinProbability(word);

        for(unsigned i=0;i<N;i++){
            double ui=0,xi=getX(i,condition);
            if (event.getEvent(i,0)==1) {
                ui=-pi_prime_divby_pi(xi);
            }
           else {
                ui=pi_prime_divby_one_minus_pi(xi);
           }

           ui=ui*Pjoin*
              *((1-pow(unit_leak,D-getTau(i,condition)))/
               \((1-unit_leak)*getSigma(i,condition)));
           GIdkl[i]+=ui;
         }
    }
  /* computing the gradient of the pressure */
    for(RasterBlockIterator& k = grammarBMS->reset();
       k.hasNext();){
         const RasterBlock& wordBMS = k.getItem();
```

```
            RasterBlock eventBMS;
            eventBMS.resetSubSequence(wordBMS,D,1);
            RasterBlock conditionBMS;
            conditionBMS.resetSubSequence(wordBMS,0,D);
            double PjoinBMS=
                 =grammarBMS->getJoinProbability(wordBMS);

            for(unsigned i=0;i<N;i++){
               double uiBMS=0,xiBMS=getX(i,conditionBMS);
               if (eventBMS.getEvent(i,0)==1) {
                    uiBMS=-pi_prime_divby_pi(xiBMS);
               }
               else {
                    uiBMS=pi_prime_divby_one_minus_pi(xiBMS);
               }
               uiBMS=uiBMS*PjoinBMS*
                *((1-pow(unit_leak,D-getTau(i,conditionBMS)))/
                /((1-unit_leak)*getSigma(i,conditionBMS)));
               GIdkl[i]-=uiBMS;
            }
     }
    /** Normalizig gradient **/
    double norm=0;
    for(unsigned int i = 0; i < N; i++){
         norm+=pow(GIdkl[i],2);
    }
    norm=sqrt(norm);
    sys::echo("\nIn gradDiv: norm=%lg",norm);
    sys::echo("\n Gradient \n");
    for(unsigned int i = 0; i < N; i++){
         if (norm > 1)
            GIdkl[i]/=norm;
         printf("  %lg  ", GIdkl[i]);
     }
    sys::echo("\n");
    return GIdkl;
}
```

and the set up of the input current according to (4.58) reads:

```
BMSPotential& BMSPotential::setCurrentDivergence
    (RasterBlockGrammar* grammar,RasterBlockGrammar*
    grammarBMS,unsigned int R){

     double *G=
       =getGradientDivergenceCurrent(grammar,grammarBMS,R);
     uint N=number_of_units;

     for (uint i=0;i<N;i++)
          unit_currents[i]+=0.1*G[i];

     delete []G;
     return *this;
}
```

We implemented the two methods from the minimization of the Kullback-Leibler divergence in order to set up synaptic weights and input current during simulation steps.

5.2.1 Remarks

Once we have implemented the functions in C++ in order to extrapolate synaptic weights and input current, we need the Kullback-Leibler divergence between approximated and real distributions to converge to zero. Thus, for minimizing effectively this measure of probability distance and reach desired synaptic weights and input current, one has to generate random initial values for these parameters and repeat to call the two main functions implemented above for a "big enough" number of iterations. We have to prove the converge to zero of KL divergence when increasing this number.

We will see in the next section simulations that Neuromathcomp team provided in order to reach a complete knowledge of the behavior of BMS model changing parameters and increasing time length of the spike trains. Further, we will discuss the convergence of the method implemented in section 5.2 through several simulations to set model parameters and to find the number of iterations required to reach the convergence.

Chapter 6

Numerical simulations

We report here some numerical simulations which has been conducted for setting firstly the model parameters ($\gamma,\ \sigma_B,\ I_e,\ N$) in order to obtain suitable conditional probabilities and a small Kullback-Leibler divergence between empirical and modeled probability measures, and secondly simulations for proving the convergence of the method developed in sections 4.5 and 5.2.

Before that, for sake of completeness it is better to explain theoretically the variability of the BMS dynamics as a function of the parameters, in agreement with [31].

6.1 BMS dynamics

6.1.1 Space phase (without noise!)

Since $\gamma < 1$ and $\sigma_B = 0$, we can define the phase space for $\boldsymbol{V}$ as a set $\mathcal{M} = [V_{\min}, V_{\max}]^N$ such that

$$V_{\min} \leq V_i(k) \leq V_{\max} \quad \forall i = 1, \ldots, N, \tag{6.1}$$

where

$$V_{\min} = \min\left(0, \frac{1}{1-\gamma}\left[\min_{i=1,\ldots,N} \sum_{j|W_{ijd}<0} W_{ijd} + I_i^{ext}\right]\right) \tag{6.2}$$

and

$$V_{\max} = \max\left(0, \frac{1}{1-\gamma}\left[\max_{i=1,\ldots,N} \sum_{j|W_{ijd}>0} W_{ijd} + I_i^{ext}\right]\right) \tag{6.3}$$

where the notation $j|W_{ijd} < 0$ ($j|W_{ijd} > 0$) specifies that all the weights taken into account are negative (positive).

Thus, for each neuron we can decompose the interval $\mathcal{I} = [V_{\min}, V_{\max}]$ into $\mathcal{I}_0 \cup \mathcal{I}_1$ with $\mathcal{I}_0 = [V_{\min}, \theta)$ and $\mathcal{I}_1 = [\theta, V_{\max}]$. If $V_i \in \mathcal{I}_0$ the neuron i is quiescent, otherwise it fires. Further, this partition permits us to define the *spiking state*, which can be expressed as an N dimensional binary vector, $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_N\} \in \Omega$. Then,

$$\mathcal{M} = \bigcup_{\boldsymbol{\omega} \in \Omega} \mathcal{M}_{\boldsymbol{\omega}} \tag{6.4}$$

where

$$\mathcal{M}_{\boldsymbol{\omega}} = \{\boldsymbol{\omega} \in \mathcal{M} | V_i \in \mathcal{I}_{\omega_i}\}\,. \tag{6.5}$$

This allows us to classify the membrane potential vectors according to their spiking state. The coefficient $\gamma(1 - Z[V_i])$ in (4.6) corresponds to contraction in direction i. Namely, suppose $V_i \in \mathcal{I}_1$, which means that $Z[V_i] = 1$, by consequence the contraction coefficient will be zero and the membrane potential reseted, otherwise the membrane potential will be contracted by a factor γ. This effect is shown in figure 6.1b and 6.1a respectively.

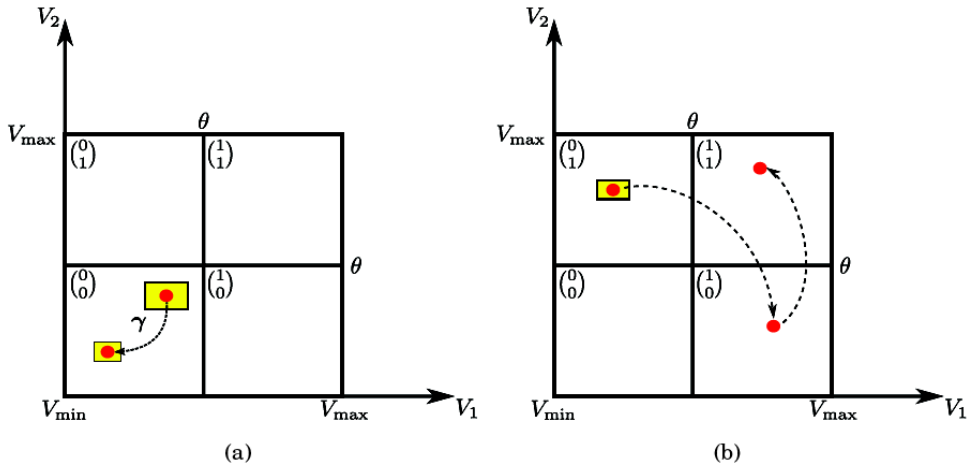


Figure 6.1: Two examples of the partion space for non-perturbed balls in a system of two neurons. The phase is partitioned from the firing state of the neurons and is labeled as $\boldsymbol{\omega} = \binom{\omega_1}{\omega_2}$. [31]

6.1.2 Singularity set

The set

$$\mathcal{S} = \{\boldsymbol{V} \in \mathcal{M}, |\exists i, V_i = \theta\} \tag{6.6}$$

is called the *singularity set* and is the set of membrane potential vectors such that at least one of the neurons has membrane potential exactly equal to the threshold.

Let us consider the trajectory of a point $\boldsymbol{V} \in \mathcal{M}$, which has perturbations with an amplitude $< \epsilon$, this is equivalent to consider the evolution of a ϵ-ball $\mathcal{B}(\boldsymbol{V}, \epsilon)$ under (4.6). Then, we have two cases:

- **The non-critical case**: when $\mathcal{B}$ does not intersect $\mathcal{S}$, $\mathcal{B}(\boldsymbol{V}, \epsilon) \cap \mathcal{S} = 0$. In this scenario the perturbation have not a considerable effect on the trajectory of $\boldsymbol{V}$ and it behaves as the previous case (fig 6.1)
- **The critical case**: when $\mathcal{B}$ intersects $\mathcal{S}$, $\mathcal{B}(\boldsymbol{V}, \epsilon) \cap \mathcal{S} \neq 0$.
  Unlike the previous case, in this scenario the crossing of $\mathcal{S}$ induces a strong effect: when $\mathcal{B}$ and $\mathcal{S}$ are intersected the trajectory of $\boldsymbol{V}$ changes drastically, since the perturbed trajectory induces a new spiking state.

The critical case is represented in figure (6.2).

6.1.3 Asymptotic dynamics

Mathematical studies in [8] have proved the existence of three main regimes in the asymptotic dynamics of BMS model for a finite size. These regimes are denoted as: *neural death, periodic* and *chaotic* regimes.

The Neural Death corresponds to a regime where neurons stop to fire due mainly to a weak activity in their synapses or a weak external current. This definition assumes that $I_i^{ext} < (1 - \gamma)\theta$ and considers the set $\mathcal{M} = \{\boldsymbol{V} | V_i < \theta, \forall i\}$.

The Periodic one corresponds to trivial activities where the neurons present repetitive patterns and occurs in the domain $\mathcal{M}_1 = \{\boldsymbol{V} | V_i \geq \theta, \forall i\}$.

The Chaotic regime occurs, as well as the periodic, in the domain $\mathcal{M}_1 = \{\boldsymbol{V} | V_i \geq \theta, \forall i\}$, but where the period tends to be very large and the behavior is in between the neural death and the periodic.

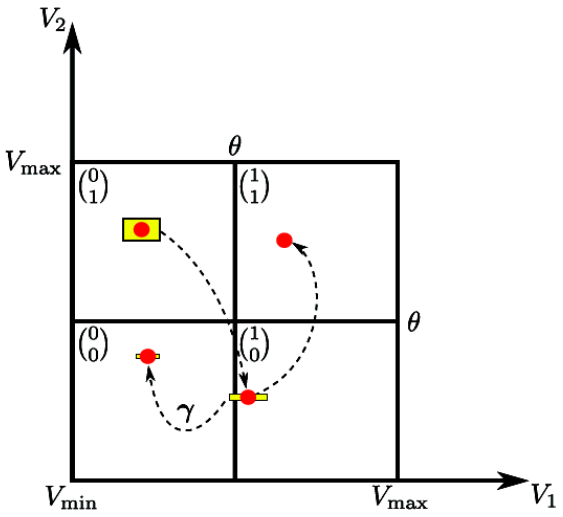


Figure 6.2: An example of the partitioned space for perturbed ϵ-balls in a system of two neurons.

The last regime is the less trivial and essentially the more interesting for studying not predictable evolution. For very small σ_B's the dynamics are almost deterministic and it is possible to recreate all the dynamical regimes. Figure 6.3 shows the period and the chaotic regimes.

Furthermore, we studied the evolution of the Kullback-Leibler divergence $d_{KL}(\mu^{(R)}, \pi^{(T)})$ as a function of T for several values of $R,\ \sigma_B,\ \gamma$ and with an external current of the form $I_k^{ext} = 0.7 \sin^2\left(\frac{2\pi k}{N}\right)$ as shown in figure (6.13).

For each set of parameters $T,\ R,\ \sigma_B,\ \gamma$, 48 networks where generated and we average d_{KL} out of these samples; this also provides error bars. Example curves are plotted in fig. 6.5. Note that the plot is in logscale. In some cases the error bars were larger than the average value: this explains why some error bars hit the x-axis.

We observe, for large enough R, that $d_{KL}(\mu^{(R)}, \pi^{(T)}) \to 0$ as $T \to +\infty$, as expected. However, the convergence rate depends on $\gamma,\ \sigma_B,\ R$. Moreover, increasing R does not necessarily improve the numerical results. The reason is that, the larger R, the larger the phase space to sample (2^{NR} blocks). Thus, for $N = 8,\ R = 4$ the phase space has 2^{32} blocks. In fact, many of these blocks have a small probability and do not contribute significantly to the KL divergence. Nevertheless, one needs large times rasters to sample cor-

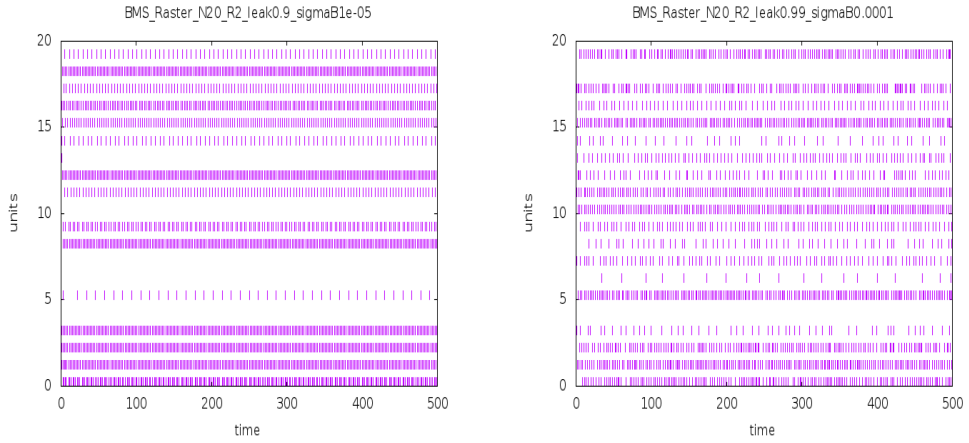


Figure 6.3: Fig (left) Periodic orbit and (right) Chaotic regime, both generated by BMS potential simulations.

rectly the phase space. When one multiplies the range by 2, as shown in the simulations, one increases the phase space dimension by 2^{NR} and so, rapidly, one is not able to generate sufficiently large rasters to sample correctly the phase space. This effect is enhanced by increasing σ_B. Indeed, generating typical rasters one sees that for small σ_B the dynamics is essentially periodic.

To better understand the quality of the approximation we add here the plots of the conditional probabilities (fig 6.6). For several values of γ, σ_B and ranges R we plotted, for each observed block ω_0^D in abscissa the theoretical probability $P\left[\omega(D)|\omega_0^{D-1}\right]$ and in ordinate the empirical average of blocks $\pi_\omega^{(T)}\left[\omega(D)|\omega_0^{D-1}\right]$ computed on an empirical raster ω of size T. Thus, each block is represented by a point in this graph. The closer this point to the diagonal, the best is the estimation. However, finite size sampling produces errors. In figure 6.6 we have compared the empirical probability of blocks observed in a raster generated by BMS model, to the theoretical probability. We have done this for the same value of γ, σ_B as in fig. 6.5 where a small range potential ($R = 4$) provides already a good approximation. For $\sigma_B = 0.1,\ 0.2$ the results are relatively good with a spread of points around the diagonal where the spreading increases with the error made on the estimation of empirical probabilities. The spread increases with σ_B, as expected since, the larger the noise, the more uniform is the probability of blocks, the larger must be T in order to have a good estimate of the empirical probabilities.

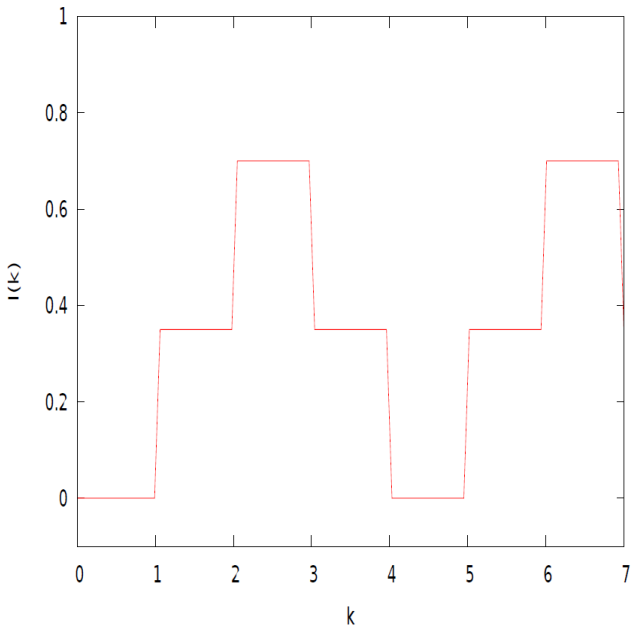
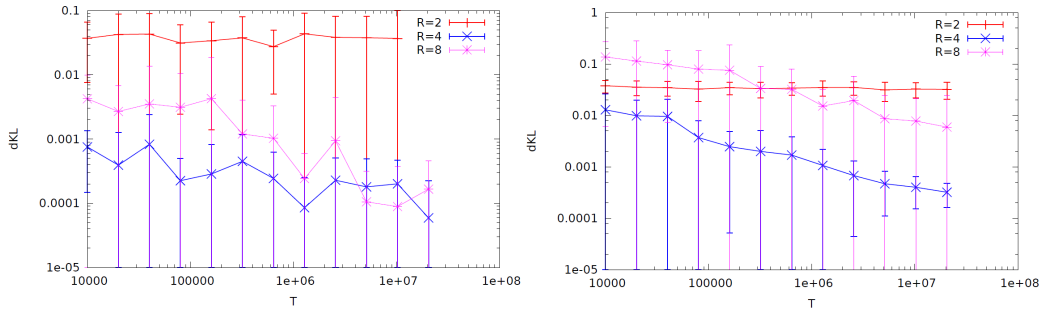


Figure 6.4: Current profile for N=8.

To quantify the amount of error on the graph we used color plots.

To summarize, we use the approximation made in section 4.4.3 of a finite range potential. For example, it is clear that $R = 2$ does not give a correct approximation. Thus, in order to achieve a good numerical fit of the model within a reasonable time, one needs to consider a small leak γ, which affords to consider a low range approximation (but not too low), and a moderate $\sigma_B \leq 0.2$ in order to limit fluctuations of empirical probabilities.

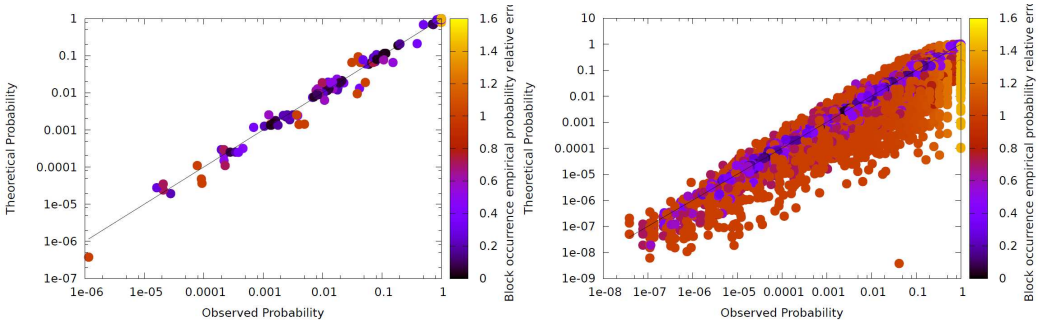
Thus, from now on we will use $R = 4$, which allows us to keep $T \leq 10^7$. Indeed, simulations indicate that T is the parameter that influences most the computational time. Since our purpose is to study the dynamics increasing size of the network of neurons in order to evaluate better the experimental data from real retina, we will simulate systems increasing the number of neurons N.



(a) $\gamma = 0.1,\ \sigma_B = 0.1$ (b) $\gamma = 0.1,\ \sigma_B = 0.2$

(c) $\gamma = 0.2,\ \sigma_B = 0.1$ (d) $\gamma = 0.2,\ \sigma_B = 0.2$

Figure 6.5: Plot of the Kullback-Leibler divergence of a function of T for $N = 8$



(a) $\gamma = 0.1,\ \sigma_B = 0.1$ (b) $\gamma = 0.1,\ \sigma_B = 0.2$

(c) $\gamma = 0.2,\ \sigma_B = 0.1$ (d) $\gamma = 0.2,\ \sigma_B = 0.2$

Figure 6.6: Plot of empirical conditional probability for $T = 10^7,\ R = 4$ and $N = 8$.

6.2 Convergence of the minimization of KL divergence with respect to synaptic weights

In order to prove the convergence of the method based on minimization of KL divergence (section 4.5), we generate two raster plots with the BMS model. The first one is at the place of a real experimental raster and we indicate it as the sought raster. The second one is the initial condition for the raster that will be change during simulation steps, since we apply the method of minimization of the KL divergence (between this and the sought raster probabilities). The method modifies the synaptic weights and the input current in the BMS model that generates this second raster. We want this raster to approximate the first one and we denote it as the approximated raster. Thus, in any simulation step we will have the *sought raster* with fixed BMS model parameters and the *approximated raster* which changes from step to step because of different BMS model parameters set by the minimization method.

Thus, we can start effective simulations. We generate randomly the "real" weights to be approximated and every time we call the function *setWeights-Divergence*, we generated the BMS raster which will be modified by the function. Then, we have to decide how to initialize the matrix of synaptic weights to give as a parameter of the model the first time we start the simulation. Since we do not know a priori which one is the best, we made different trials with different initial weights matrices to ensure the independence on initial conditions. We used weights of the form: a matrix only of zero values, a matrix in which all elements are equal to the elements of the sought weights matrix and finally one of random elements.

Our purpose is to verify that the output weights, after a suitable number of iterations, converge to the real sought weights (that we generate as well with BMS model). Thus, to evaluate this effect, we plot the mean difference between the approximated weights and the sought weights, we call it "distance", for each iteration as follows:

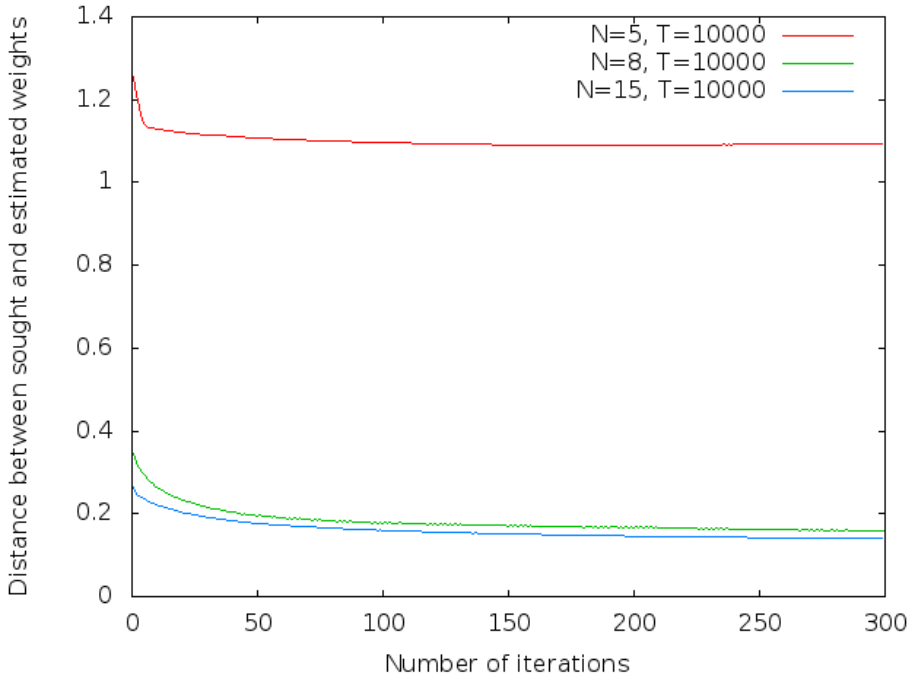
$$\text{Distance}(t) = \frac{1}{N^2} \sum_{i,j} \left| W_{ij}^{ext}(t) - W_{ij}^{real} \right|, \tag{6.7}$$

where

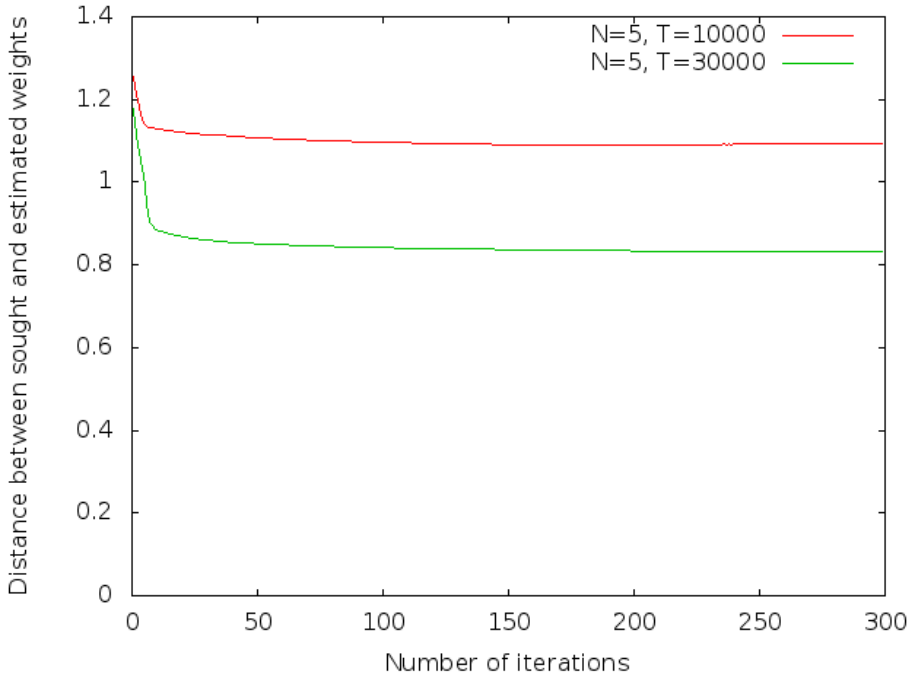
$$W_{ij}^{ext}(t) = W_{ij}^{ext}(t-1) + \frac{dW_{ij}^{ext}}{dt}, \tag{6.8}$$

with $W_{ij}^{ext}(0)$ the initial value for all $i,\ j = 1, \ldots, N$. We observed for different initial weight matrices more or less the same trend and we report in figures 6.7 the more significant, the one with randomly generated weights.

These two figures show examples of distance changing in fig (a) the number of neurons or the time length of the rasters in fig (b). It is clear that the distance converges and from figure 6.7b we can note that increasing T the convergence limit decreases. Nevertheless, it seems it does not converge exactly to zero as we expected. In this previous simulation we used an $\epsilon = 0.5$ for 4.54, thus to improve the converge we can try to make different trials with different ϵ's and we checked if the KL divergence of the estimated raster decreases with the distance. We plot in figures 6.8 the KL divergence between the sought raster and approximated raster probabilities for three different values of ϵ. The KL divergence is calculated every time our method *setWeightsDivergence* is called, thus it changes in any iteration and we expect it to converge to zero.

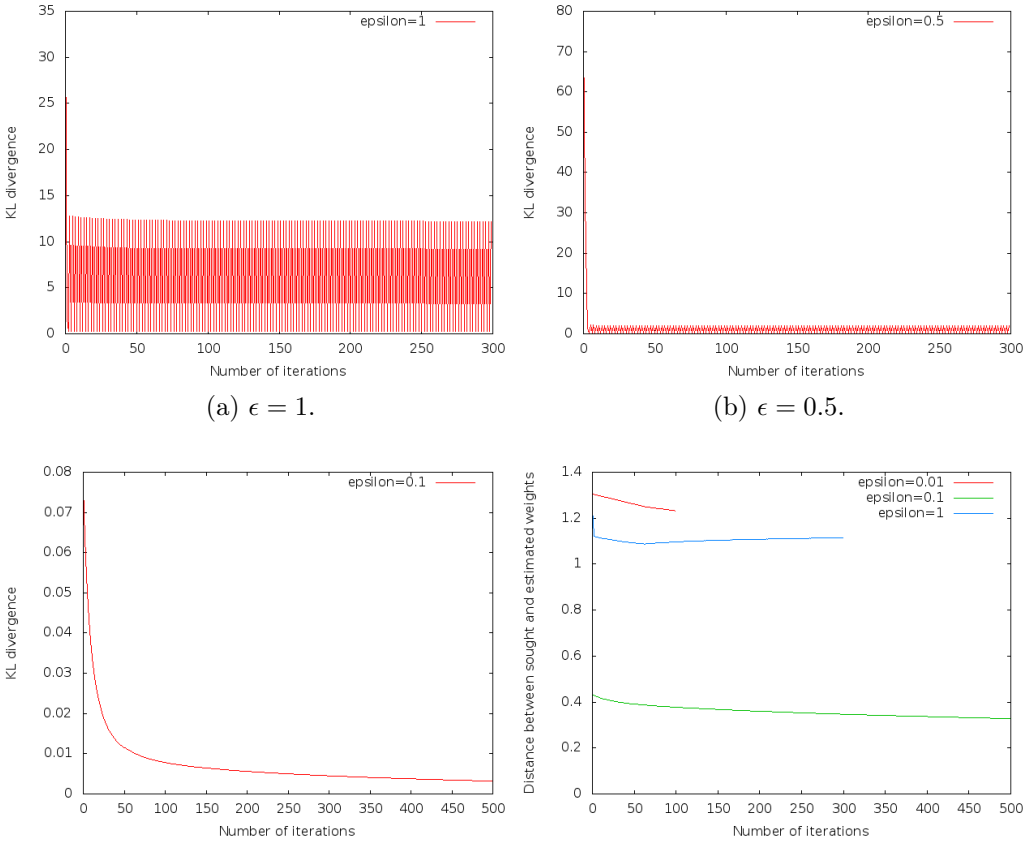


(a) Changing number of neurons.



(b) Changing the time lenght of the raster T.

Figure 6.7: The mean difference (distance) between approximated weights and real weights for $R = 4,\ \gamma = 0.2,\ \sigma_B = 0.2$. We use here random initial weights and $\epsilon = 0.5$.



(a) $\epsilon = 1$.

(b) $\epsilon = 0.5$.

(c) $\epsilon = 0.1$.

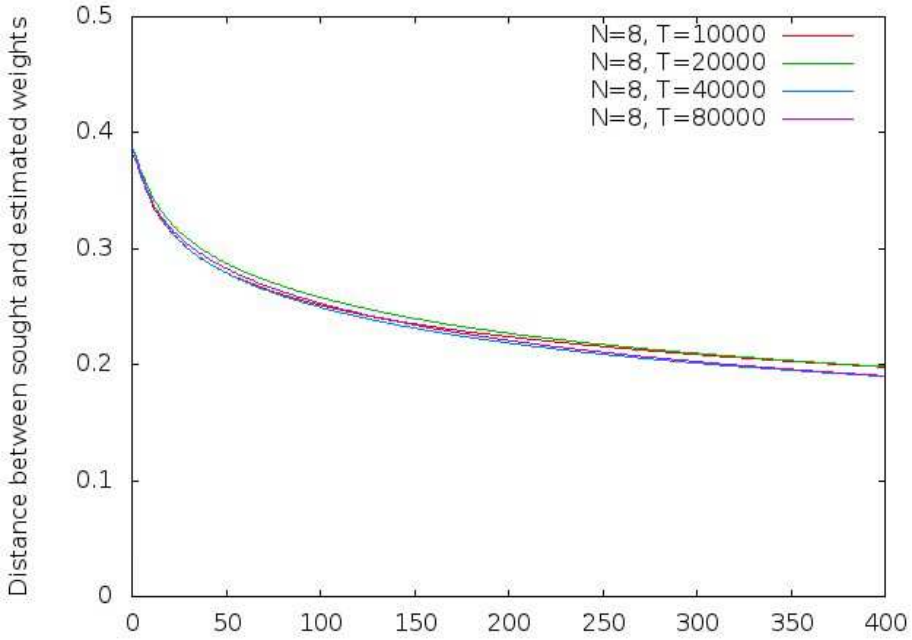
(d) The distance between sought and estimated weights for different ϵ.

Figure 6.8: Plots of KL divergence in (a), (b) and (c) and in (d) the distance between weights, for $N = 5,\ R = 4,\ \gamma = 0.2,\ \sigma_B = 0.2$ and $T = 10^5$ changing different ϵ's

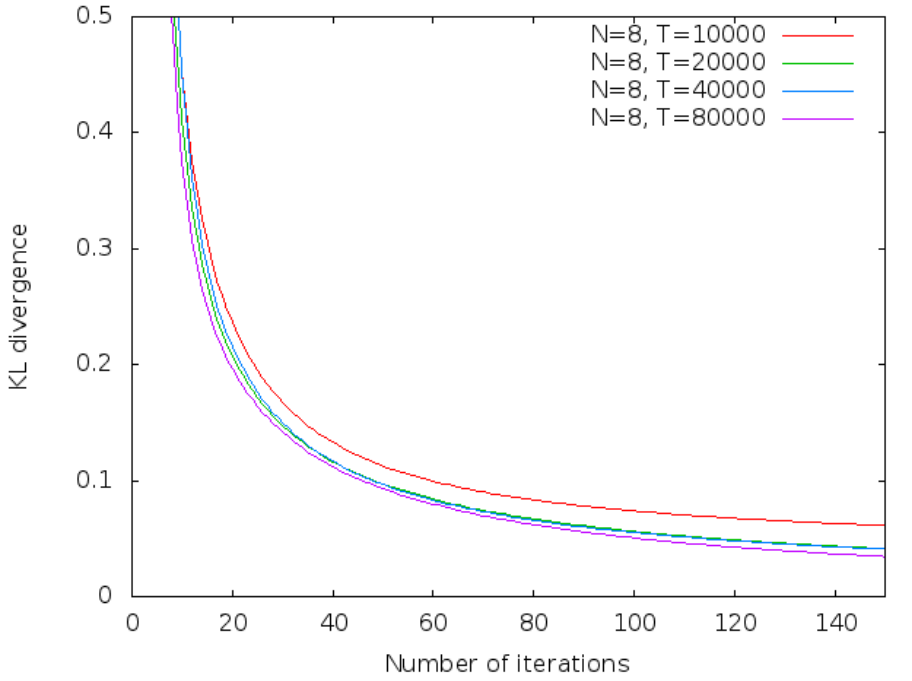
From figure 6.8 we observe that the choice of $\epsilon = 0.1$ arises naturally. For this value the KL divergence converges to zero without fluctuations and the limit of the distance is nearer to zero. It is important to underline that the choice of taking into account the KL divergence is not casual. First of all the implemented method needs the KL divergence to go to zero for its definition, than this quantity is the most important one in order to approximate the statistics of rasters, since for example it could be possible to have different synaptic weight matrices with the same probability measure. Furthermore, when we will show simulations with experimental data from real retina, KL divergence will be the only index we have to check the goodness of our method.

Another encouraging factor is the improvement in convergence of the distance and KL divergence when the length T increases as it was in plot 6.5 for the empirical probability. We can see it from figures 6.9 which show for "standard" model parameters the trend of the distance and the KL divergence.

Having set the more important parameters as ϵ and the number of iterations, we evaluate the distance averaged over randomly generated initial weights, maintaining fixed the real sought weights. In addition, we computed the variance of this quantity (error bars in the plot) to better evaluate it, figures 6.10 and 6.11.



(a) Distance between sought and estimated weights changing T.



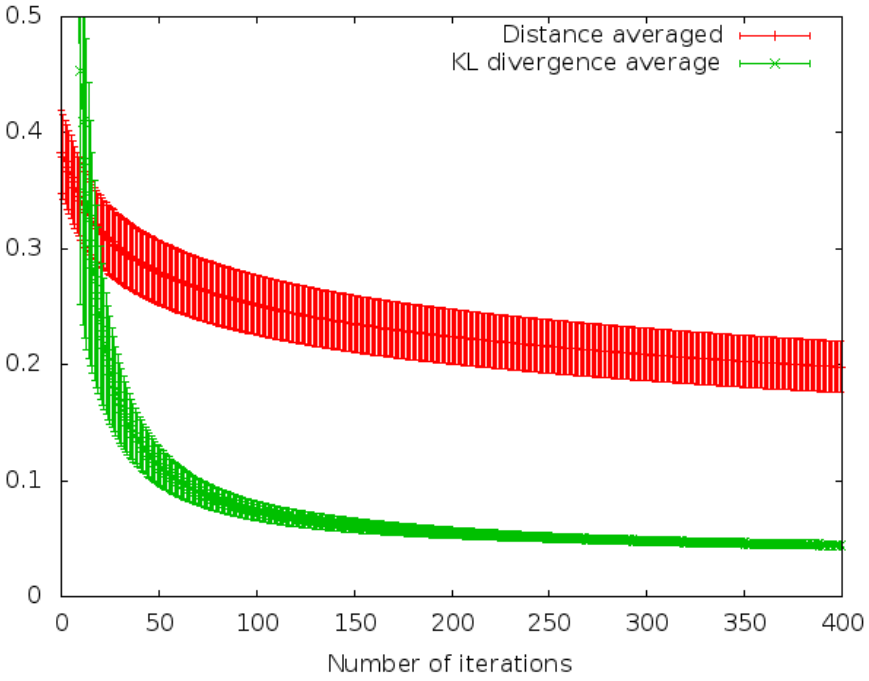
(b) KL divergence changing T.

Figure 6.9: Plots for $R = 4,\ \gamma = 0.2,\ \sigma_B = 0.2,\ \epsilon = 0.1$ and $N = 8$

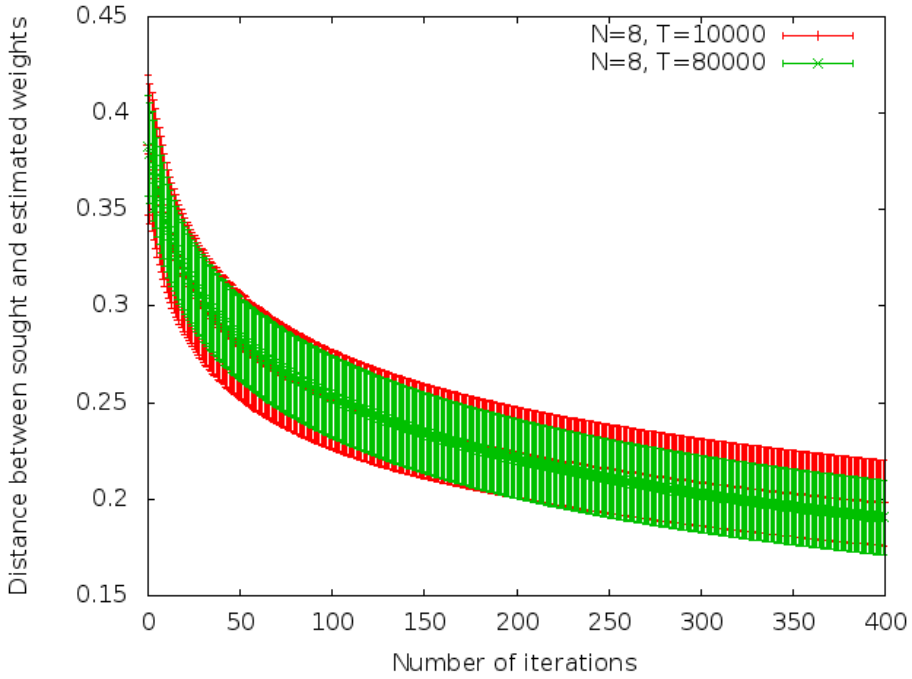
We would like to obtain similar results as before, namely to obtain the convergence of the divergence and the averaged distance. We add here the code we used to averaged on initial weights for sake of completeness:

```
const unsigned int Nmax=300;  //Number of iterations
const unsigned int Nsamp=50;
                //Number of different initial matrix
                   weights
for(unsigned int j=0;j<Nsamp;j++){
                //generation of initial weight loop
     BMSPotential bms_test;
     bms_test.reset(N, leak, sigmaB, Ie);
     bms_test.setWeights(sigma, bias);
     printf("\n%d-th initial weights. \n", j+1);
     PrintWeights(bms_test);
     RasterBlock *raster =
         =bms_test.getRasterBlock(transients,T);
     RasterBlockEmpiricalProbability Prob;
     Prob.reset(*raster);
     double rate=Prob.getAverageEmpiricalRate();
     for(unsigned int i=0; i<Nmax; i++) {
                                    //Iteration loop
         double k=0;
         sys::echo("\n\nWeights computation");
         bms_test.setWeightsDivergence(&grammar,R);
         sys::echo("done");
         double dkl=bms_test.getDivergence(raster,R);
         sys::echo("done");
         k=L1distance(bms_test,bms_sought);
         diff_moy[i]+=k;
         diff_var[i]+=pow(k,2);
     }
  }
for(unsigned int i=0;i<Nmax;i++){
     diff_mean[i]/=(double)Nsamp;
     diff_variance[i]=sqrt(((diff_var[i]/(double)Nsamp)-
     -pow(diff_moy[i],2))*((double)Nsamp/
     /(double)(Nsamp-1)));
}
```

The results with the same parameters as above are in agreement with previous results, as we can see in figures 6.10 and 6.11.



(a) KL divergence and distance averaged for $T = 10^5$.



(b) Distance averaged between sought and estimated weights changing T.

Figure 6.10: Plots for $R = 4$, $\gamma = 0.2$, $\sigma_B = 0.2$, $\epsilon = 0.1$ and $N = 8$, both the quantity averaged on 50 different matrices of initial weights.

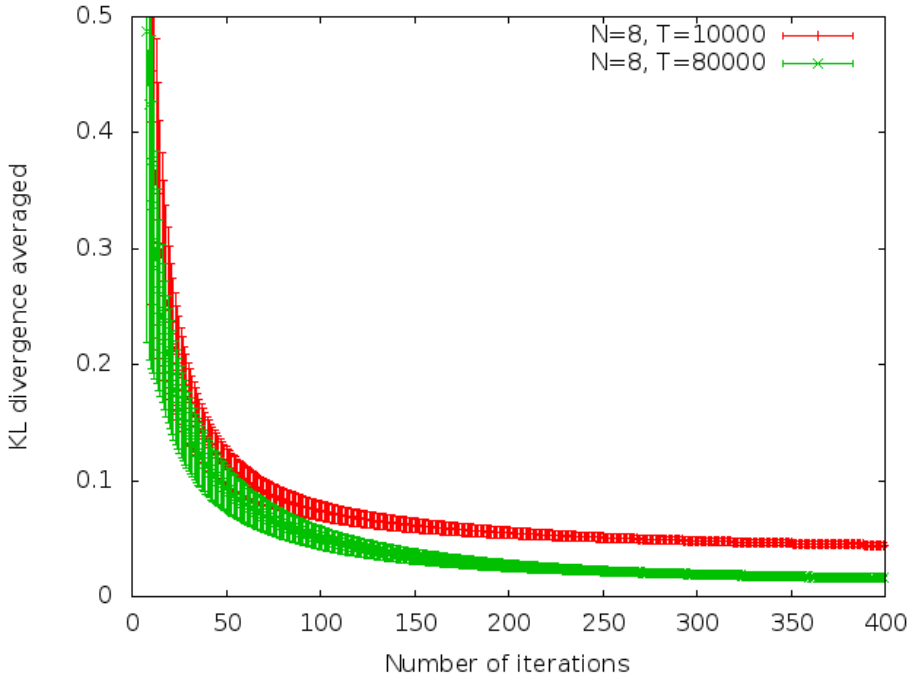


Figure 6.11: Plots of the KL divergence for $R = 4$, $\gamma = 0.2$, $\sigma_B = 0.2$, $\epsilon = 0.1$ and $N = 8$, averaged on 50 different matrices of initial weights.

6.3 Convergence of the minimization of KL divergence with respect to input current

We repeated the same simulations as previous paragraph applied to the input current. The KL divergence and the distance with respect to input current seems to converge to zero even better the KL and distance with respect to the weights, as we can see from figures 6.13.

Thus, we can compare for the same model parameters the KL divergence between probabilities of the sought raster and the extrapolated approximated raster where only synaptic weights are derived from the minimization method, and the KL divergence between probabilities of the sought raster and the approximated raster where both synaptic weights and input current are derived from minimization methods. We can see from figure 6.14 that, as we expected, the convergence adding the method for extrapolating the current input improves.

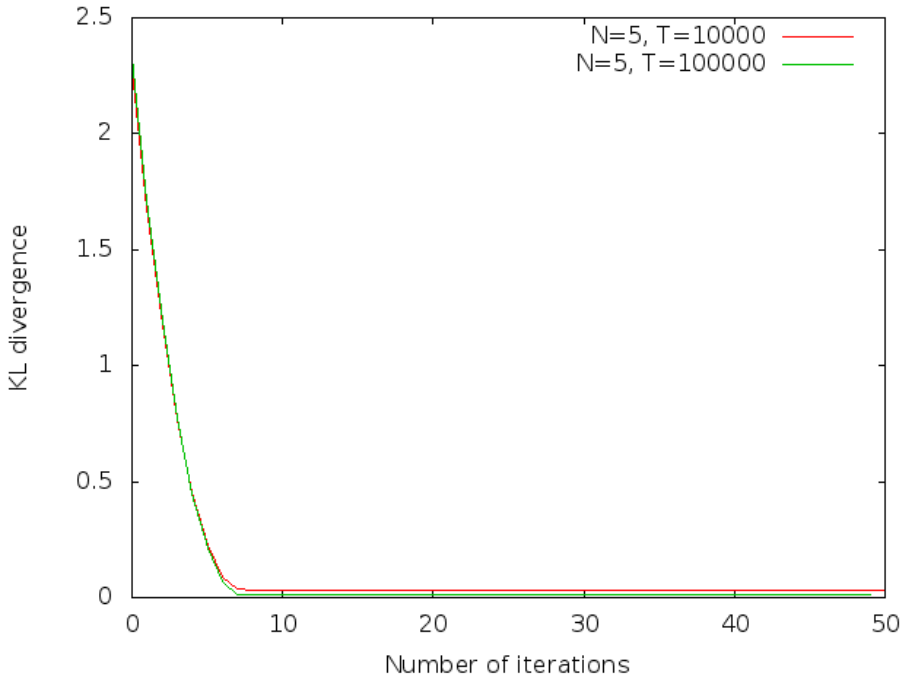
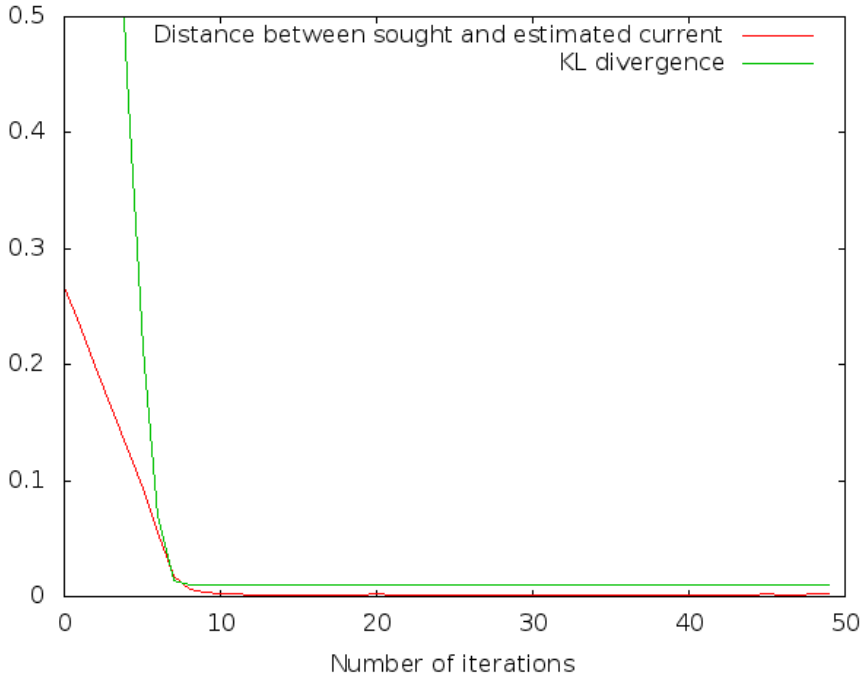
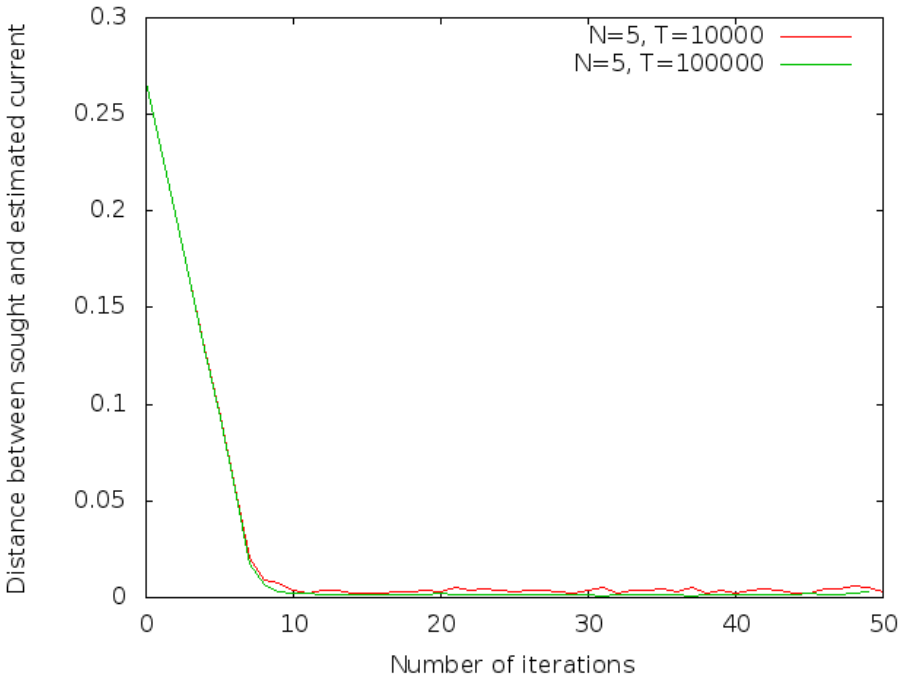


Figure 6.12: Plots of the KL divergence for $R = 4$, $\gamma = 0.2$, $\sigma_B = 0.2$, $\epsilon = 0.1$ and $N = 5$ with input current updated with minimization method.



(a) KL divergence and distance between sought and estimated currents for $N = 5$ and $T = 10^5$.



(b) Distance between sought and estimated currents changing T.

Figure 6.13: Plots for $R = 4,\ \gamma = 0.2,\ \sigma_B = 0.2,\ \epsilon = 0.1$ and $N = 5$

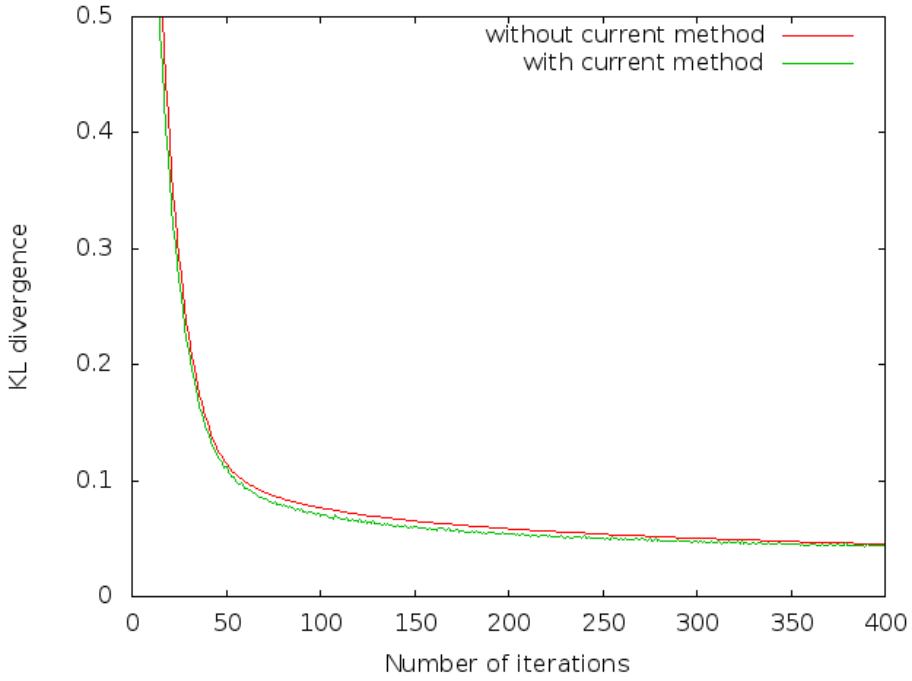


Figure 6.14: Plots of the KL divergence for $R = 4$, $\gamma = 0.2$, $\sigma_B = 0.2$, $\epsilon = 0.1$ and $N = 5$. The green curve is with the current updating and the red one is without.

6.4 Experimental data from a real retina

We applied our methods to some experimental data taken from a retina in vitro at which it is shown a movie of a moving fish.

The result in figure 6.15 we obtain after simulations is not yet fully satisfying. Nevertheless, we have to take into account that this is only a preliminary trial where we used σ_B, γ and other parameters by chance without knowing the real ones. Furthermore, in these simulations only 20 neurons over hundreds present in the data file are exploited, that the reason why we are not so surprised to find a convergence not perfectly to zero. If we apply both the methods of minimization with respect to the weights and with respect to the input current together, we can see a considerable improvement in the convergence than before (fig 6.16), even though it is not converging to zero for the reasons explained above.

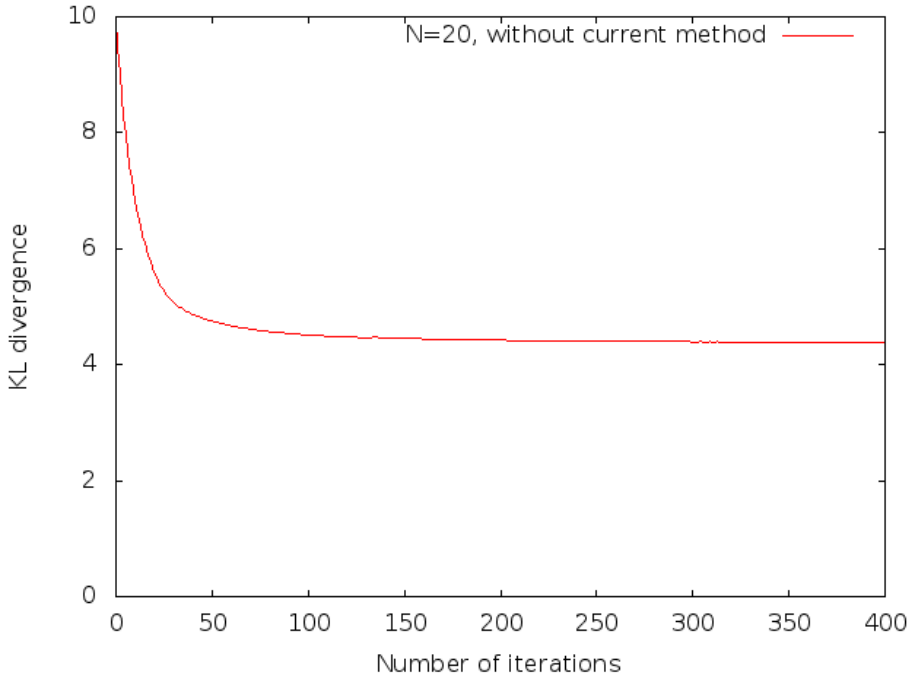


Figure 6.15: Plots of the KL divergence only with minimization with respect to the weights for experimental data in retina, with $R = 4$, $\gamma = 0.2$, $\sigma_B = 0.2$, $\epsilon = 0.1$ and $N = 8$. The time length of the real raster is $T = 10818$

It is worth to underline that to reach perfect condition in simulations on real experimental data it will be taken months to study the nature of data and to set the best the parameters for that data. This is only a preliminary application to real spikes from retina in order to understand if the method is conducting us in the right direction, and results from figure 6.16 are quite encouraging.

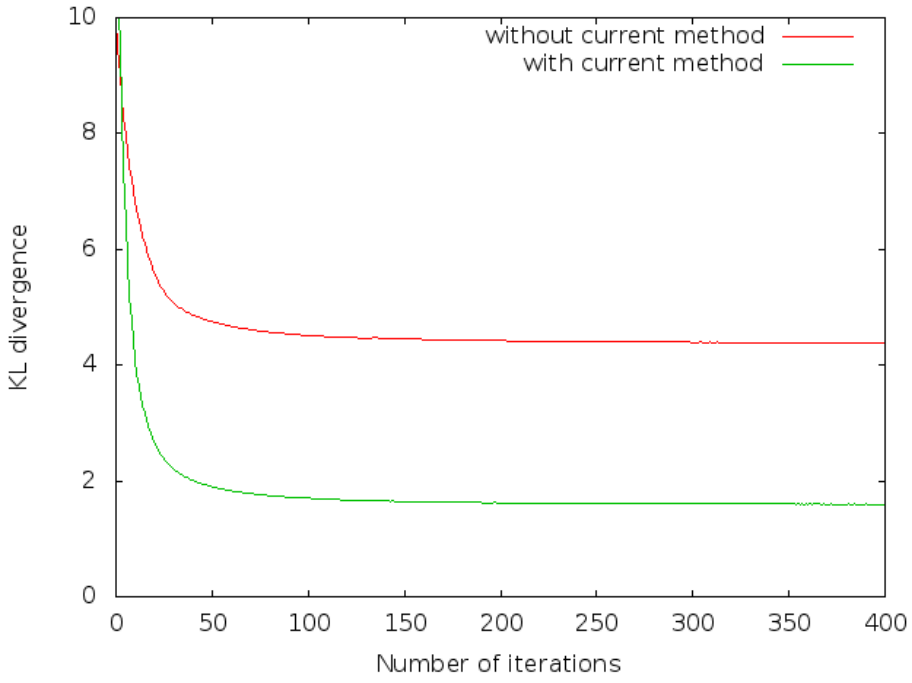


Figure 6.16: Plots of the KL divergence only with minimization with respect to the weights (red) and minimization with respect to the weights and current (green) for experimental data in retina, with $R = 4,\ \gamma = 0.2,\ \sigma_B = 0.2,\ \epsilon = 0.1$ and $N = 8$. The time length of the real raster is $T = 10818$

Chapter 7

Conclusions and perspectives

In this thesis, a statistical analysis of sequences of action potential emitted by a network of neurons is discussed. This analysis is based on the attempt to infer the hidden distribution of these spike trains from data with the help of the Kullback-Leibler (KL) divergence as a measure of the distance between probabilities. In chapter 3, we illustrated all the essential mathematical and statistical tools useful to understand the theory behind this manuscript. According to thermodynamics formalism [9] in statistical mechanics, the network is studied as a macroscopic system where neurons are considered as microscopic particles which can assume only two values 1 or 0 if respectively the neuron fires or not. From the ergodic theory, the Gibbs distribution of Markov Chains which maximizes the entropy is used to approximate the hidden probability and the degree of the approximation can be control with KL divergence [37] [6] [5].

Taking into account the difficulties in obtaining a statistical model directly from experimental data (controlling the parameters or having "clean" data) or in selecting one model among many others or other complications such as finite sampling effects, adaptation mechanism, non-stationarity and more, it appeared easier [3] [2] to characterize spike trains statistics in neural networks models where one controls exactly the neural network parameters, the number of involved neurons, the number of samples, and the duration of the experiment. In chapter 4, the BMS model, a particular example of Leak Integrate-and-fire model (LIF) was defined [31] [3].

In the present thesis, a novel contribution to the existing literature is the method developed in section 4.5, where we made use of the minimization of the KL divergence between the "real" and the approximated distributions

to fit parameters of the model such as synaptic weights and input current. We extrapolate the derivative of KL divergence with respect to synaptic weights and with respect to input current in order to measure time variations of these two parameters. Basically, we add a temporal dependence of the parameters that we do not have before in works such as [3] [4] where all model parameters are set at the begin of simulations. From biophysical point of view, the temporal dependence is widely recognized to be essential for information processing in brain. It has been studied [1] [30] [2] [12] [14] [16] [19] [22] that synaptic plasticity, which is responsable of learning and memory, changes in time the connections between neurons and takes into account the spiking timing of neurons. Despite being far from providing a complete model to mimic STDP (spike time dependent plasticity) or Hebbian learning (mechanisms explained in sections 2.3 and 2.4), our proposed formulation introduces a temporal component into a previously developed model [3] [4]. In chapter 5, we reported the C++ code of BMS model's functions developed by INRIA researchers and the implementation of the updating of synaptic weights and input current thanks to the minimization of KL divergence.

Finally, in chapter 6 we tried the method applying it first to spike trains generated artificially by the BMS model and than to real experimental data taken from a retina in vitro. We observed in the first case that the KL divergence goes effectively to zero when the length of spike sequences increases, whereas for experimental data the converge to zero is more difficult to reach even increasing greatly the length of sequences. This is mostly due to the fact that we apply the method without knowing anything about some right parameters to fix in the model and we take them by chance. Moreover, we considered only 20 neurons over hundreds present in the data file since the duration of simulations was already too long to handle. However, we believe this thesis to be a small step forward that needs to be explored and pursue more deeply. It would be interesting to continue to investigate the method for experimental data of different nature, for different parameters model and further splitting the C++ code in parallel processes in order to carry out simulations that involve a lager number of neurons. Moreover, another viable option to investigate could be to apply the proposed method to different neuron models largely discussed in literature, such as generalized Integrate-and-Fire [8] or Fitzhugh-Nagumo model [13].

Bibliography

[1] L. F. Abbott and P. Dayan. *Theoretical Neuroscience*. The MIT press, 2001.

[2] A. Burkitt, J. L. Van Hemmen, and M. Gilson. Spike-timing-dependent plasticity for neurons with recurrent connections. *Biological Cybernetics*, 96:533–546, 2007.

[3] B. Cessac. A discrete time neural network model with spiking neurons. *Journal of Mathematical Biology*, 56:311–345, 2008.

[4] B. Cessac. Statistics of spike trains in conductance-based neural networks : Rigorous results. *Journal of Mathematical Neuroscience*, 1:1–42, 2011.

[5] B. Cessac and R. Cofré. Spike train statistics and Gibbs distributions. *Journal of physiology, Paris*, 107:360–368, November 2013.

[6] B. Cessac and A. Palacios. Spike Train Statistics from Empirical Facts to Theory: The Case of the Retina. In *Modeling in Computational Biology and Biomedicine*, pages 261–302. Springer Berlin Heidelberg, 2013.

[7] B. Cessac and J. C. Vasquez. Parametric estimation of spike train statistics by Gibbs distributions: an application to bio-inspired and experimental data. In *Cinquième conférence plénière française de Neurosciences Computationnelles, "Neurocomp'10"*, volume 1, pages 1–5, 2010.

[8] B. Cessac and T. Viéville. On dynamics of integrate-and-fire neural networks with conductance based synapses. *Frontiers in computational neuroscience*, 2:2, 2008.

[9] J. Chazottes and G. Keller. *Pressure and equilibrium states in ergodic theory*. Springer New York, 2011.

[10] G. Gallavotti, F. Bonetto, and G. Gentile. *Aspects of ergodic, qualitative and statistical theory of motion*. Springer-Verlag Berlin Heidelberg, 2004.

[11] L. Gatteschi. *Funzioni speciali*. Unione tipografico-editrice torinese, 1973.

[12] J. P. Gavornik, M. G. H. Shuler, Y. Loewenstein, M. F. Bear, and H. Z. Shouval. Learning reward timing in cortex through reward dependent expression of synaptic plasticity. *Proceedings of the National Academy of Sciences of the United States of America*, 106(16):6826–6831, 2009.

[13] W. Gerstner. Integrate-and-fire neurons and networks. *The handbook of brain theory and neural networks*, (1998):1–12, 2002.

[14] W. Gerstner and W. M. Kistler. *Spiking Neuron Models single neurons, populations, plasticity*. Cambrige University Press, 2002.

[15] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. M. Herz. Neural codes: firing rates and beyond. *Proceedings of the National Academy of Sciences of the United States of America*, 94(24):12740–12741, November 1997.

[16] M. Gilson. STDP in recurrent neuronal networks. *Frontiers in computational neurosciencecomputational neuroscience*, 4:1–15, 2010.

[17] M. Gilson, A. N. Burkitt, D. B. Grayden, Thomas D. A., and J. L. Van Hemmen. Emergence of network structure due to spike-timing-dependent plasticity in recurrent neuronal networks IV Structuring synaptic pathways among recurrent connections. *Biological Cybernetics*, 101:427–444, 2009.

[18] M. Gilson, T. Masquelier, and E. Hugues. STDP allows fast rate-modulated coding with Poisson-like spike trains. *PLoS computational biology*, 7(10):e1002231, 2011.

[19] R. Gütig, R. Aharonov, S. Rotter, and H. Sompolinsky. Learning Input Correlations through Nonlinear Temporally Asymmetric Hebbian Plasticity. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 23(9):3697–3714, 2003.

[20] D. Heeger. Poisson model of spike generation. *Handout, University of Standford*, pages 1–13, 2000.

[21] E. M. Izhikevich, J. A. Gally, and G. M. Edelman. Spike-timing dynamics of neuronal groups. *Cerebral cortex*, 14(8):933–944, 2004.

[22] H. Markram, W. Gerstner, and P. J. Sjöström. A history of spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 3(August):1–24, 2011.

[23] R. H. Masland. Cell populations of the retina: the Proctor lecture. *Investigative ophthalmology & visual science*, 52(7):4581–4591, June 2011.

[24] T. Masquelier and G. Deco. Learning and Coding in Neural Networks. In *Principles of Neural Coding*, chapter 26, pages 513–526. CRC Press, 2012.

[25] A. Mohemmed and S. Schliebs. SPAN : Spike Pattern Association Neuron for Learning Spatio-Temporal Spike Patterns. *International Journal of Neural Systems*, 22(4):1–16, 2012.

[26] B. Nessler, M. Pfeiffer, and W. Maass. STDP enables spiking neurons to detect hidden causes of their inputs. *Advances in Neural Information Processing Systems*, 22:1357–1365, 2010.

[27] W. Parry and M. Pollicott. *Zeta functions and the periodic orbit structure of hyperbolic dynamics*. Asterisque, 1990.

[28] C. Pouzat and A. Chaffiol. On goodness of fit tests for models of neuronal spike trains considered as counting processes, 2009.

[29] D. S. Reich, F. Mechler, K. P. Purpura, and J. D. Victor. Interspike intervals, receptive fields, and information encoding in primary visual cortex. *The Journal of neuroscience*, 20(5):1964–1974, March 2000.

[30] B. Romain. Generation of correlated spike trains. *Neural computation*, 21:188–215, 2009.

[31] H. Rostro-gonzález. *Computing with spikes, architecture, properties and implementation of emerging paradigms*. PhD thesis, 2011.

[32] E. Schneidman and M. Berry. Weak pairwise correlations imply string correlated network states in a neural population. *Nature*, 440:1007–1012, 2006.

[33] J. Shlens. Notes on kullback-leibler divergence and likelihood theory. *Systems Neurobiology Laboratory*, 92037:1–4, 2007.

[34] J. Shlens, G. D. Field, J. L. Gauthier, M. I. Grivich, D. Petrusca, A. Sher, A. M. Litke, and E. J. Chichilnisky. The structure of multi-neuron firing patterns in primate retina. *The Journal of neuroscience*, 26(32):8254–8266, 2006.

[35] S. Song, K. D. Miller, and L. F. Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, September 2000.

[36] F. G. Tricomi. *Fonctions hypergéométriques confluentes*. Gauthier-Villars, 1960.

[37] J. C. Vasquez and B. Cessac. How Gibbs Distributions May Naturally Arise from Synaptic Adaptation Mechanisms. A Model-Based Argumentation. *Journal of Statistical Physics*, 136(6):565–602, 2009.

[38] J. C. Vasquez, B. Cessac, and T. Vieville. Entropy-based parametric estimation of spike train statistics. Technical report, INRIA Research Report, 2010.

[39] A. Wohrer and P. Kornprobst. Virtual Retina: a biological retina model and simulator, with contrast gain control. *Journal of computational neuroscience*, 26(2):219–249, 2009.