Analysis of large scale spiking networks dynamics with spatio-temporal constraints : application to multi-electrodes acquisitions in the retina

Hassan Nasser

PhD THESIS

prepared at

INRIA Sophia-Antipolis

and presented at the

University of Nice Sophia-Antipolis
Doctoral School of Information and Communication Sciences

A dissertation submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF SCIENCE

Specialized in Control, Signal and Image Processing

Analysis of large scale spiking networks dynamics with spatio-temporal constraints: application to Multi-Electrodes acquisitions in the retina

Hassan NASSER

Advisor	Pr. Bruno Cessac	INRIA Sophia-Antipolis
Reviewers	Pr. Stefano Panzeri	Instituto Italiano de Technologia (visio-conference)
	Pr. Simona Cocco	Laboratoire de Physique Statistique de l'ENS
Examiners	Pr. Pierre Kornprobst	INRIA Sophia-Antipolis
	Pr. Adrián Palacios	University of Valparaíso, Chile (visio-conference)
	Dr. Matthias Hennig	University of Edinburgh
	Dr. Olivier Marre	Institut de la vision, Paris, France

UNIVERSITÉ DE NICE SOPHIA-ANTIPOLIS - UFR SCIENCES

École Doctorale STIC

Sciences et Technologies de l'Information et de la Communication

THÈSE

Pour obtenir le titre de

DOCTEUR DE SCIENCES

de l'université de Nice Sophia-Antipolis

Discipline: Automatique, Traitement du Signal et des Images

présentée et soutenue par

Hassan NASSER

Analyse des trains de spike à large échelle avec contraintes spatio-temporelles: application aux acquisitions Multi-Electrodes rétiniennes

Thèse dirigée par Pr. Bruno CESSAC

Rapporteurs	Pr. Stefano Panzeri	Instituto Italiano de Technologia (visio-conférence)
	Pr. Simona Cocco	Laboratoire de Physique Statistique de l'ENS
Examinateurs	Pr. Pierre Kornprobst	INRIA Sophia-Antipolis
	Pr. Adrián Palacios	University of Valparaíso, Chile (visio-conférence)
	Dr. Matthias Hennig	University of Edinburgh
	Dr. Olivier Marre	Institut de la vision, Paris, France

Contents

List of Figures

Acknowledgments

"Gratitude is when memory is stored in the heart and not in the mind."
Lionel Hampton.

First and foremost I express my sincerest gratitude to my supervisor, Prof. Bruno Cessac, who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my PhD to his encouragement and effort and without him this thesis, would not have been completed or written. One simply could not wish for a better or friendlier supervisor. Bruno, the boss, the friend and the coach.

Special thanks to the Examiners, Prof. Simona Cocco and Prof. Stefano Panzeri, who not only honored me to be a part of the jury, but also for the time they committed to read my thesis, give me precious comments, advices and support.

I also gratefully acknowledge Prof Adrian Palacios, Dr. Olivier Marre and Dr. Matthias H Hennig who accepted to be a part of the jury. I thank them also for their advice, support and comments at several stages of my thesis.

Special thanks to Prof. Oliver Faugeras who welcomed me in his team. I greatly acknowledge his leadership and scientific attitude, and also the friendly vibes he ***emits*** whenever he is.

I would like to thank also the sponsors of this work: KEOPS and REN-VISION European projects who provided the scholarship over 3.5 years, the materials as well as the travels fees for participating in international conferences and visiting the CINV (Chile) for 3 weeks.

No way to pass this acknowledgments without sending great thanks to Prof. Pierre Kornprobst who accepted my application to the computational biology program 4 years ago. I consider that my career life changed dramat-

ically because of this master program. Not only at the educational level, but the fact that this master made me discover new city, new life, new environment: this changed my life. Prof. Kornprobst also was the president of my Jury and I greatly thank him for his precious participation and comments on the presentation and manuscript.

I would like also to thank the University of Nice Sophia Antipolis, whom together with INRIA created an excellent research environment. I thank especially the president Mme Frédérique Vidal. I would like also to thank the team of the UNICE foundation who honored me with the best prize in the Innovative Startup Competition (2013 edition). Special thanks to the director Mme Caty Conraux who never hesitated to encourage me and showed always her interests in my project.

I wish to thanks my colleagues who became very good friends. All the people I met during my stay at INRIA, those who moved from INRIA and those who are still there. I won't take the risk to name them all: I already wrote their names in my heart.

My very closed friends, in France and overall the globe: Manal, Diab, Caroline, Houssein, Rami, Ali, Samer, Sami, Ibrahim. If you are not with me now, I know that I am in your heart. Thank you very much for your support.

My parents, Hana and Charif, and my sisters and brothers: the greatest thanks ever. Your love provided my inspiration and was my driving force. I owe you everything and wish I could show you just how much I love and appreciate you. I hope that this work makes you proud.

Finally, to the martyrs who passed away in my native country, in south Lebanon: thank you for defending and protecting us against terror. We all owe you our lives; without you, we wouldn't be still alive. You (martyrs) are really the candles that burns itself to light the way of others (us). I exclusively offer my work for you.

Abbreviations

MEP	Maximum Entropy Principle
MaxEnt	Maximum Entropy
MEA	Multi Electrode Array
Ntimes	The number of Montecarlo samples
Nflips	The number of time we toggle the state of events in a Gibbs Sample
KLD	Kullback-Leibler Divergence
CLT	Central Limit Theorem

List of symbols

$\omega_i(n)$	Spike event
$\omega(n)$	Spike pattern
$\omega_{n_1}^{n_2}$	Spike block
ω	Spike train
T	Length (in time) of the spike train
N	Number of neurons
R	Model range
D	Model memory ($R = D - 1$)
$m_l(\omega)$	Monomial number l
$\mathbf{m}$	Vector of monomials
L	Total number of parameters (monomials) in the model
λ_l	Parameter number l
$\boldsymbol{\lambda}$	Parameters vector
$\mathcal{H}$	Gibbs potential
$Z_{\boldsymbol{\lambda}}$	Partition function
$\mathcal{S}$	Entropy
$\mathcal{P}$	Topological pressure
$\pi_\omega^{(T)}$	Empirical probability measured on the spike train, ω, of length T
$\mu_{\boldsymbol{\lambda}}$	Gibbs density with parameters $\boldsymbol{\lambda}$
$\mathcal{M}$	Set of invariant probabilities
$\delta_l = \lambda_l' - \lambda_l$	Learning rate or the value by which we update the parameters, λ_l
$\boldsymbol{\delta}$	Vector of learning rates
d_{KL}	Kullback-Leibler divergence
C_{jk}	Correlation between two monomials, j and k
χ	Hessian matrix (second derivative of the pressure)
Δ	Root sum square of the learning rates
$\boldsymbol{\beta}$	Fluctuations on the monomials averages
$\boldsymbol{\epsilon}$	Fluctuations on the parameters (relaxation)

Abstract (English)

Recent experimental advances have made it possible to record up to several hundreds of neurons simultaneously in the cortex or in the retina. Analyzing such data requires mathematical and numerical methods to describe the spatio-temporal correlations in population activity. This can be done thanks to Maximum Entropy method. Here, a crucial parameter is the product $N \times R$ where N is the number of neurons and R the memory depth of correlations (how far in the past does the spike activity affects the current state). Standard statistical mechanics methods are limited to spatial correlation structure with $R = 1$ (e.g. Ising model) whereas methods based on transfer matrices, allowing the analysis of spatio-temporal correlations, are limited to $NR \leq 20$. In the first part of the thesis we propose a modified version of the transfer matrix method, based on the parallel version of the Montecarlo algorithm, allowing us to go to $NR \approx 100$. In a second part we present EnaS, a C++ library with a Graphical User Interface developed for neuroscientists. EnaS offers highly interactive tools that allow users to manage data, perform empirical statistics, modeling and visualizing results. Finally, in a third part, we test our method on synthetic and real data sets acquired from retina and provided by neuroscientists partners. Our non extensive analysis shows the advantages of considering spatio-temporal correlations for the analysis of retina spike trains, but it also outlines the limits of Maximum Entropy methods.

Résumé (Français)

L'évolution des techniques d'acquisition de l'activité neuronale permet désormais d'enregistrer simultanément jusqu'à plusieurs centaines de neurones dans le cortex ou dans la rétine. L'analyse de ces données nécessite des méthodes mathématiques et numériques pour décrire les corrélations spatio-temporelles de la population neuronale. Une méthode couramment employée est basée sur le principe d'entropie maximale. Dans ce cas, le produit $N \times R$, où N est le nombre de neurones et R le temps maximal considéré dans les corrélations, est un paramètre crucial. Les méthodes de physique statistique usuelles sont limitées aux corrélations spatiales avec $R = 1$ (Ising) alors que les méthodes basées sur des matrices de transfert, permettant l'analyse des corrélations spatio-temporelles ($R > 1$), sont limitées à $N \times R \leq 20$. Dans une première partie, nous proposons une version modifiée de la méthode de matrice de transfert, basée sur un algorithme de Monte-Carlo parallèle, qui nous permet d'aller jusqu'à $NR \approx 100$. Dans la deuxième partie, nous présentons la bibliothèque C++ Enas, dotée d'une interface graphique développée pour les neurobiologistes. Enas offre un environnement hautement interactif permettant aux utilisateurs de gérer les données, effectuer des analyses empiriques, interpoler des modèles statistiques et visualiser les résultats. Enfin, dans une troisième partie, nous testons notre méthode sur des données synthétiques et réelles (rétine, fournies par nos partenaires biologistes). Notre analyse non exhaustive montre l'avantage de considérer des corrélations spatio-temporelles pour l'analyse des données rétiniennes; mais elle montre aussi les limites des méthodes d'entropie maximale.

Chapter 1

Introduction

Contents

"Neurons are discrete and autonomous cells that can interact." Santiago Ramón Y Cajal, 1906.

1.1 Introduction

In this chapter we introduce the problematic we addressed. We begin with an introduction to neurons without going into details because we do not consider the neurons activity at a microscopic level (ions exchange, membrane potential, etc ...) but only at a behavioral level (did the neuron spike or not!). Afterwards, we briefly introduce the wide field of neural coding and we precise in which niche this work fits. Finally, we present our goal and the chapters organization.

1.2 Neurons speak "spikes"

Sensory neurons (Figure 1.1(a)) respond to stimulus and **transmit** corresponding information to the brain. A stimulus induces changes in the voltage across the neuron's membrane, which, when it reaches a certain threshold, it produces a signal: action potential. The action potential has always the same shape for a given neuron. This fact made researchers substitute an action potential with a "spike", a vertical dash that expresses if a neuron fires or not.

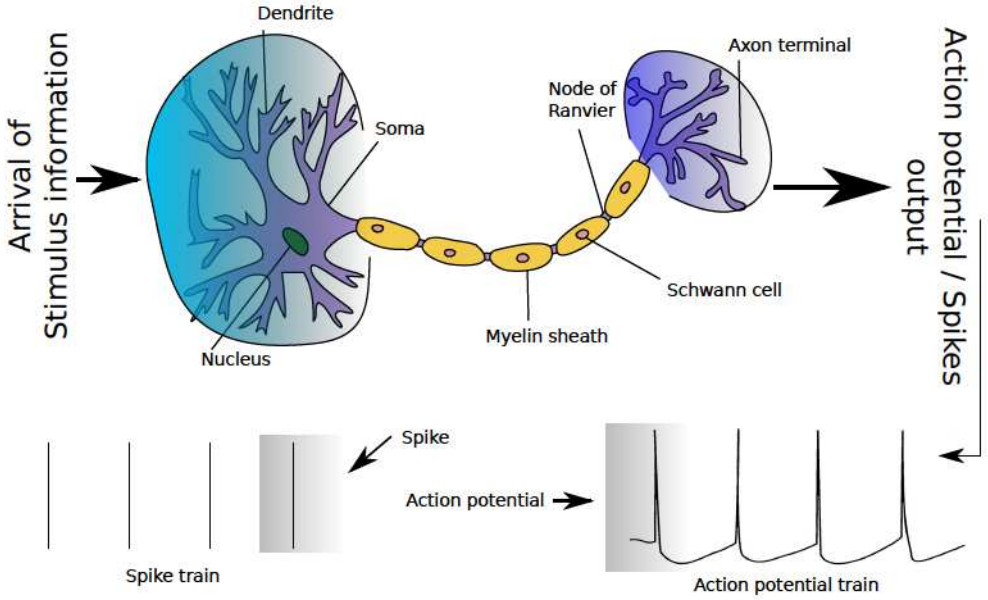
Neurons also **interact** with each other and **communicate** information using spikes. So, this is their language. If one wants to study the information transmitted by neurons, one has to study the spatio-temporal sequences of spikes emitted by neurons: spike trains. The science of studying this information is an emergent domain and called neural coding. In order to situate our work, we give a brief introduction about the main axis of this field in the next section.

The retina (Figure 1.1(b)) is one of the most challenging studied neural networks because of its importance as a sensory organ as well as the possible bio-inspired technologies (retinal prosthetic, robotics, image processing) that benefits from the understanding of the retinal functioning.

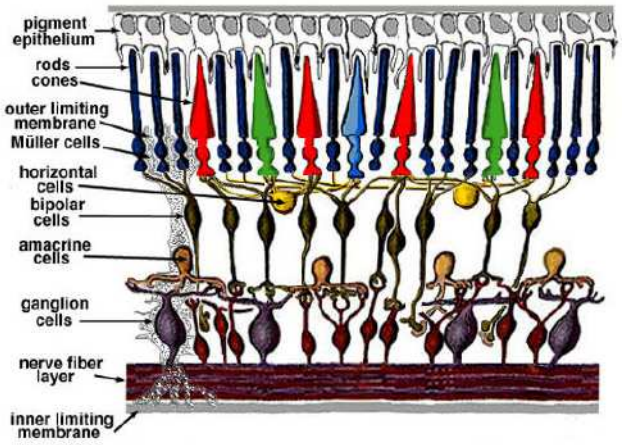
1.3 Neural coding

One of the fundamental problems in systems neuroscience is that of determining the functional relationship between the stimulus and the spike responses of neurons. This problem referred to as the "neural coding problem". It is like the dictionary "stimulus(S) $\rightleftarrows$ response(R)". Neural coding exists in two modes: encoding and decoding. Encoding is mapping from stimulus to response. Decoding is the inverse problem: what is the stimulus given some response? Figure 1.2 shows an example of neural coding. This example is taken from [Gollisch and Meister, 2008] and illustrates the relation between the stimulus that a retina receives and the response of the retinal ganglion cells. We observe a systematic relation between the stimulus (here a gratings with several spatial phases) and two measurable quantities in the response of those cells (here spike count and the latency). That is, the retina translates the spatial phase of the grating into spike count and latency code. The neural coding exists in several schemes:

- Rate coding: it assumes that information about the stimulus is contained in the firing rates.



(a) The neuron and the spike.



(b) The retina.

Figure 1.1: (a) The neuron receives the information about the stimulus via the dendrites terminals, processes it in the Soma and then transmits information in the form of action potentials. As shown, there is a train of action potentials. We substitute them with spikes or bits of information to make processing easier. Modified from `wikipedia.org`. (b) The retina is system of 5 cell layers (photo-receptors, bipolar cells, horizontal cells, amacrine cells and retinal ganglion cells (RGC)). The photo-receptors transduce light to electrical signals. The RGC transmit the information to the brain through the optic nerve. RGC and other cells process the information before sending it to the brain. Source: `wikispaces.com`.

- Spike count rate: also called temporal average. It consists in counting the number of spikes during a single trial of the stimulus (usually of length 500-1000 ms).

- Time-dependent firing rate: it consists on studying how the firing rate evolves over the spike train. This techniques is also used to measure the stationarity of data.

- Temporal coding: this technique assumes that the information is contained in the precise time of spikes. For example, measuring the time of the first spike after stimulating cells.

- Population coding: it considers that the information is coded in the joint activity of neurons (such as interactions).

A detailed description on the neural coding schemes is investigated in [Aldworth et al., 2011], [Panzeri et al., 2010]. In this thesis we consider the population coding scheme using a probabilistic modeling approach based on the Maximum Entropy Principle.

1.3.1 $S|R$ relation and response variability

Nothing guarantees that the response R will be the same if the stimulus S is repeated on several trials. In fact, at each trial, we observe a variability in the response of the retina. Figure 1.3 shows the response an RGC over 18 trials of the same stimulus (data D_{11}, explained in chapter 5). We observe a variability in the response from one trial to another. We have seen also in Figure 1.2 that the rate and latency have been measured over several trials because the variability of the response. This variability is caused by noise artifacts, acquisitions systems reliability, spike sorting accuracy .... Some studies also suggested that one of the reasons for which there is a variability of the response is that there is a uniform noise in the retinal ganglion cells themselves [Croner et al., 1993]. Thus, if the response is not the same at each trial and for the same stimulus, this means that the empirical statistics that we perform on the data are not the same. For this reason, a common belief, which will be our working hypothesis, is that the response, and related sample empirical averages, are generated by a hidden probabilistic model reflecting some hidden laws and causality, that we try to infer from data. This is a key step toward deciphering the neural code: having a probabilistic model $P[S|R]$, one can infer a probabilistic "dictionary" for the code, based on the Bayes concept:

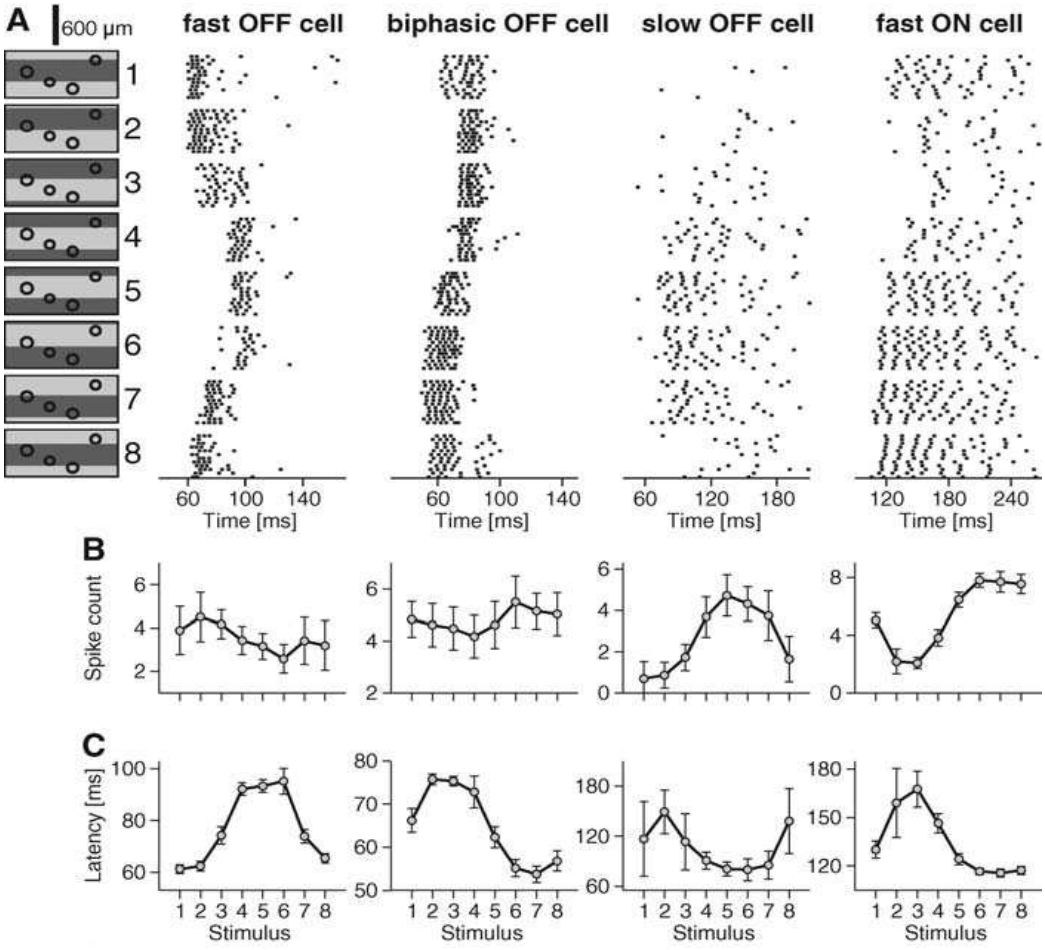


Figure 1.2: Ganglion cell responses to flashed gratings with different spatial phases. (A) Raster plots of spike responses from four ganglion cells to several 150 ms presentations of each of eight gratings. Time is measured from stimulus onset.(Left) Schematic drawings of the eight stimuli with different spatial phases. The circles show 1-SD contour lines of the spatial receptive fields of the four cells, correspondingly from left to right, in relation to the stimuli. (B) Tuning curves of the elicited spike count. Here and in subsequent figures, all error bars show the standard deviation across trials with the same stimulus. (C) Tuning curves of the first-spike latency. "Fast OFF" and "biphasic OFF" cells typically showed strong tuning in the latency and only mild tuning in spike count; despite their names, these cell types receive input from both ON and OFF pathways (19). "Slow OFF" and "ON" cells, on the other hand, displayed good tuning in the spike count and often did not respond with spikes to all stimuli. The relatively long latencies are typical for cold-blooded animals. From [Gollisch and Meister, 2008].

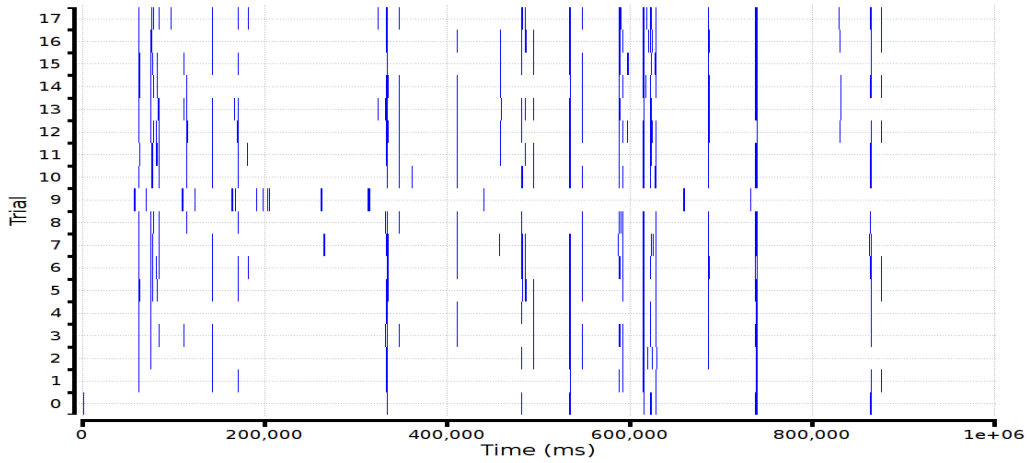


Figure 1.3: The response of 1 neuron over 18 trials. The neuron is a ganglion cell from the salamandar retina and the stimulus is a real movie (data set D_{11} explained in chapter 5).

$$\begin{aligned} P[R|S] \times P[S] &= P[S|R] \times P[R], \\ P[R|S] &= \frac{P[S|R]}{P[S]} \times P[R]. \end{aligned} \tag{1.1}$$

Our aim is to find a probabilistic model $P[R|S]$: what is the probability of having a response R if we know the stimulus S.

1.3.2 Probabilistic modeling categories

There are two main categories of probabilistic models:

- Stochastic models: those models are used to describe an undeterministic process. They consist of a collection of random variables which describes a process that evolves in an undeterministic way, independently from the starting point (which means that the future state of the system is unpredictable and may evolve in an infinite number directions). The Gaussian model, that could be described by two parameters, mean and standard deviation, is one example.

- Graphical models: Denotes the conditional dependence between the random variables. In other words, it emphasizes the fact that the state

of one random variable depends on the state of other random variable(s). The graphical models contains two main subfamilies: Bayesian networks and Markov Random Fields (MRF).

Since we are interested in studying the neural activity in a collective fashion, we will concentrate on MRF because it offers the possibility to represent the spatial and temporal dependencies between variables (which will be the spiking events).

Modeling $P[S|R]$

We want a probabilistic model that fits to the data, taking into account the following points:

- The spatial dependencies: it represent the dependence between two or more neurons at the same time -instantaneous dependence. In fact, neurons do not act independently. Correlations between neurons have been investigated in several studies and they affect the statistics, although they are very small compared to the individual activity ([Schneidman et al., 2006], [Ganmor et al., 2011a]). We can validate this assumption by showing the error committed by an independent model when we test how it predicts higher order activities (correlations, interactions with delay ...). For example, if we assume that the activity of neurons is independent, which mean that the model is governed by a Bernoulli law, the model will not be able to accurately predict the instantaneous correlations.

- The temporal dependencies: where there is a dependence between two or more neurons, within a certain delay. In other words, the retinal activity at an instant t depends on the past activity. The memory concept, which identifies how long the retina uses from its past activity in order to determine the current state, should be in our probabilistic model. In other words, the memory is one of the model characteristics. The importance of temporal dependencies has been proven to be essential for spike train statistical models in [Tang et al., 2008], [Vasquez et al., 2012].

The Maximum Entropy $P[R|S]$

The Maximum Entropy is the modeling framework we will use. It was first introduced by [Jaynes, 1957] as a method to infer probabilistic models that maximizes our knowledge about the data. This framework imposes itself as a

natural candidate because it allows one to take into account any measurable quantity (as a feature in the model) from the spike train. For instance, one can consider that the model contains only individual activities of neurons and see whether it permits to predict higher order activities such as pairwise correlations. Adding pairwise with delay is also possible, yet not trivial. Recently, [Vasquez et al., 2012] proposed a Maximum Entropy framework that accounts for temporal dependencies (the spatial and spatio-temporal models will be presented in details in the next chapter). However it allows one to study 20 neurons maximum for spatial distribution and 5-10 neurons for distribution with memory 1-2 time units. Here we propose another approach allowing to analyze larger networks.

1.4 Review of approaches distinct from Maximum Entropy

Here we shortly review several interesting approaches attempting to decipher the neural code with probabilistic models.

One approach is provided by the so-called Linear-Nonlinear (LN) models and Generalized Linear Models (GLM) [Brillinger, 1988, McCullagh and Nelder, 1989, Paninski, 2004, Truccolo et al., 2005, Pillow et al., 2005, Pillow et al., 2008, Ahmadian et al., 2011, Pillow et al., 2011]. The idea is to model spike statistics by a point process where the instantaneous firing rate of a neuron is a nonlinear function of the past network activity including feedback and interactions between neurons [Simoncelli et al., 2004]. This model has been applied in a wide variety of experimental settings [Brillinger, 1992, Chichilnisky, 2001, Theunissen et al., 2001, Brown et al., 2003, Paninski et al., 2004, Truccolo et al., 2005, Pillow et al., 2008]. Here, neurons are assumed to be conditionally-independent given the past. The probability to have a given spike-response to a stimulus, given the past activity of the network, reads as the product of firing rates (see e.g. eq. 2.4 in [Ahmadian et al., 2011]). They define a Markov process where transition probabilities are known. The main advantages of the GLM are: (i) transition probability is known (postulated) from the beginning and does not require the heavy normalization; (ii) The model parameters have a neurophysiological interpretation, and their number grows at most as a power law in the number of neurons (as opposite to the Maximum Entropy, where the parameters are delicate to interpret and whose number can become quite large, depending on the set of constraints). The main drawback of this approach is the assumption of conditional independence between neurons. Recently, Macke et al. [Macke et al., 2011] extended the GLM model

to fix the lack of instantaneous correlations between neurons in the GLM. They added a common input function that has a linear temporal dynamics. However, one of the disadvantages of this technique is that its likelihood is not unimodal, and thus that computationally expensive: Expectation Maximization algorithms have to be used to fit parameters. The GLM model is usually used to model both the stimulus-response dependence as well as the interaction between neurons, while the Maximum Entropy models usually focus on the latter (but see [Tkacik et al., 2010]).

Another approach is the one presented in [Cocco et al., 2009]. Cocco and co-workers consider retinal ganglion cells spiking activity with a dual approach: on one hand they consider an Ising model (and higher order spatial terms) where they propose an inverse method based on a cluster expansion to find efficiently the coupling in Ising model from data; on the other hand, they consider the problem of finding the parameters (synaptic couplings) in a Integrate and Fire model with noise, from its spike trains. In the weak noise limit the conditional probability of a spiking pattern given the past is given by a least action principle. This probability is a Gibbs distribution whose normalized potential is characterized by the action computed over an optimal path. This approach allows the characterization of spatio temporal events. Especially it gives a very good fit of the cross-correlograms.

Finally, [Roudi and Hertz, 2011] consider a one step memory Markov chain where the conditional probability has a time-dependent potential of Ising type. Adapting a Thouless Anderson-Palmer [Thouless et al., 1977] approach used formerly in the Sherrington-Kirkpatrick mean-field model of spin glasses [Sherrington and Kirkpatrick, 1975] they propose an inversion algorithm to find the model-parameters. As in the GLM their model assumes conditional independence given the past.

1.5 Goal of the thesis

An interesting subject in today's neuroscience is to study the neural code at a large scale and decipher the importance of temporal and spatial dependencies between neurons. This is currently not possible for large network with existing frameworks based on Maximum Entropy models with spatio-temporal constraints because of computational complexity [1]. In parallel, the Multi-Electrode array technologies are evolving dramatically and the number

[1] We measure the scale using the product $N \times (D+1)$, where N is the number of neurons and D is the memory effect taken into account in the model. We will consider that we are in the large scale case when $N \times (D+1) > 20$.

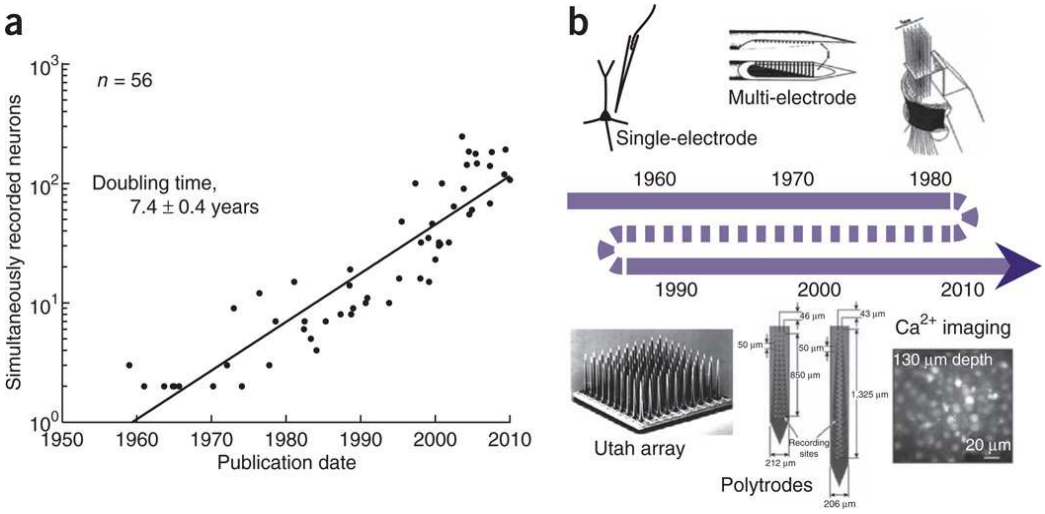


Figure 1.4: The evolution of the MEA technology over years. The number of acquired neurons double every 8 years. The last advances in MEA techniques was published in [Ferrea et al., 2012]. In 2020, we may have a 8100 channels MEA systems. Taken from [Stevenson and Kording, 2011].

of neurons we can acquire simultaneously is doubling each 8 years (Figure 1.4). Hence, we are faced with a critical mass of neural data without the possibility of studying the spatio-temporal models hidden in the activity of neurons at large scale.

The goal of this thesis is to make one step toward exploring large scale spike trains with spatio-temporal dependencies. We propose a mathematical framework and a software for this issue.

The mathematical framework is based on the transfer matrix principle but extended thanks to Montecarlo to allow analyzing large scale networks. We also propose a method to fit the parameters of the statistical models. This framework could be also a subject of study in order to improve it. One of the aims of this thesis is also to precise critical aspects of the framework and give roadmaps for further investigations. This issue is discussed in the last chapter. Finally, we also wanted to provide a first version of a software that could be interesting for researchers curious to understand neural coding. This software has been released in a first version and could also be a subject of improvement itself. In last chapter we suggest new features to add in the software.

1.6 Organization and chapters summary

This thesis is divided into 5 chapters. In the chapter 2 we present the mathematical framework for inferring Maximum Entropy models from spike trains, based on the transfer matrix technique, used to compute a probability distribution given its parameters. We will discuss the case of spatial and spatio-temporal models and give some examples used in the literature, such as Ising model. The second chapter itself is an introduction to the third chapter.

Chapter 3 shows that it is possible to use the mathematical framework explained in chapter 2 for large networks by changing the method of computing the probability distribution from a given set of parameters. We will replace the transfer matrix technique with a Montecarlo based method that allows one to compute probability distributions for larger networks. We will also show the parallelization of the Montecarlo method on multi processors computers and clusters. In the second part of this chapter we will show a method to compute the parameters. In fact, for small scale network, the Kullback-Leibler divergence could be easily computed and then we can minimize explicitly this divergence in order to fit the parameters. However, for large networks, where this divergence is hard to compute, we will use another criterion for fitting the parameters. The method is inspired from [Dudík et al., 2004] and extended to the spatio-temporal case. Overall, the chapter 2 and 3 present the mathematical framework.

Chapter 4 presents the *EnaS* library which contains the computational tools we developed to perform empirical statistics and fitting models on spike trains. This chapter will explain the main functionalities of the library. We will also explain in details some of the computational techniques we used to manage spike train data. Based on this library, we developed a highly interactive graphical user interface (GUI) that allows non-programmers to performs statistics on spike train data. We will explain this GUI as well as the task that can accomplish.

In chapter 5 we show results on real data using the computational framework we developed, *EnaS*. First, we explain briefly the Multi-Electrode acquisitions from retina dissection until spike sorting. Afterwards, we explain about the data sets that our collaborators provided and on which we performed analysis on small and large scale. We show results on those data for several models in spatial and spatio-temporal case.

Finally, chapter 6 is dedicated to discussions and perspectives. After applying our mathematical framework on the real data, we suggest several open questions that could be the subject of other projects.

Chapter 2

Modeling dynamics of spike trains with spatial and spatio-temporal constraints: a review

Contents

In this chapter we introduce the definitions we will use along the thesis (section 2.1) and then review the Maximum Entropy method for modeling spatial and spatio-temporal distributions for spike train data (section 2.2). In section 2.3 and 2.4 we explain the fitting parameters process and we give a practical example to clarify the process. Finally, we highlight the limitations of the existing methods in order to introduce the next chapter (section 2.6).

2.1 Definitions

2.1.1 Spiking objects

The spiking objects represent forms of activities in the spike train. We have 4 types of spiking objects (Figure 2.1):

- Event, represents the activity of a single neuron i at a single time t. It is defined by:

$$\omega_i(t) = \begin{cases} 1, & \text{if the neuron } i \text{ fires at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

- Spike pattern, $\{\omega_i(t)\}_{i=1,N}$, represents the activity of the whole network at a precise time t.

- Spike block, $\omega_{t_1}^{t_2} = \{\omega(t)_{t_1 \leq t \leq t_2}\}$, represents the activity of the whole network between two times t_1 and t_2.

- spike train, ω, is a set of events that represent the activity of a neural network of size N over a time period T time-steps. We assume that time has been discretized.

Spike trains are obtained after spike sorting of Multi-Electrode acquisitions (MEA) acquisitions. The MEA allows the acquisition of many signals in the same time. MEA and spike sorting are explained in details in the chapter 5 since they are not crucial to understand the following mathematical framework. Here are some examples of spike patterns and blocks on a set of 3 neurons:

- $\begin{smallmatrix}0\\0\\1\end{smallmatrix}$ is a spike pattern (of depth 1 by definition) and it means that the neuron 0 fires while other neurons are silent. $\begin{smallmatrix}1\\0\\1\end{smallmatrix}$ is also a spike pattern and it means that neurons 0 and 2 fire together and neuron 1 is silent.

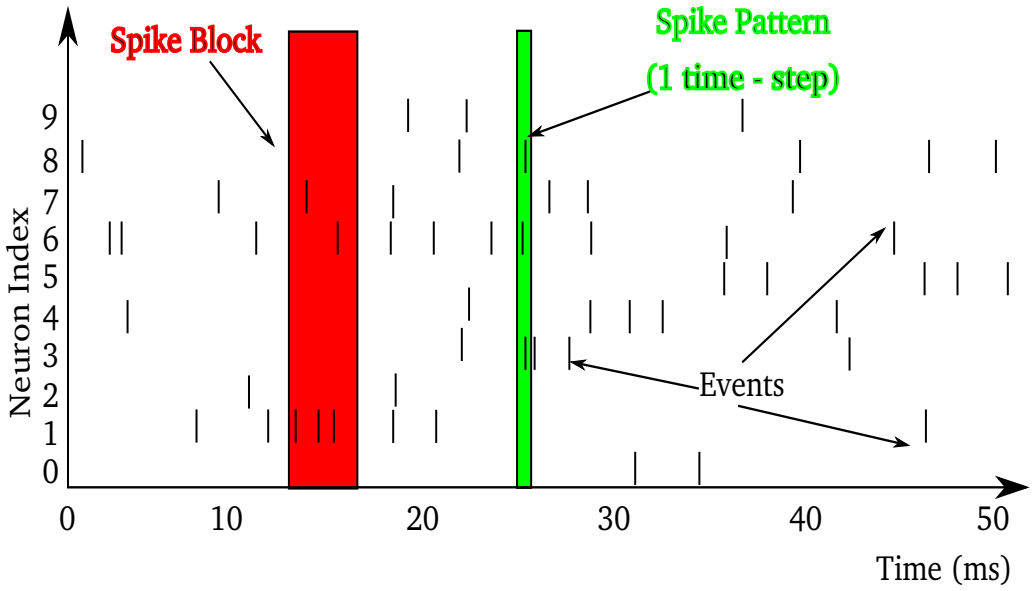


Figure 2.1: The horizontal and vertical axis represent respectively time (usually in ms) and neuron index. The spike pattern corresponds to a 1 time-step (green box). The spike block corresponds to several time-steps (red box). An event (Eq. 2.1) is defined by a neuron index and a time-stamp , (i, t) (black vertical sticks). **Note that in this thesis, we consider counting neurons beginning from 0, which is always at the bottom in the spike train graph. As we go up, we plot neurons activities with higher indexes.**

- $\begin{smallmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{smallmatrix}$ is a spike block of depth 2. It represents the fact that the neurons 0 and 2 fire together while the neuron 1 is silent, right after the neuron 1 fires and all other neurons are silent.

We can have $2^{N \times R}$ possible patterns of time range R in a network of N neurons. For instance, in a network of 3 neurons, we have $2^{3 \times 1} = 8$ possible patterns of time range 1 and $2^{2 \times 3} = 64$ patterns of depth 2.

2.1.2 Monomials

A monomial is a logical AND operator which associates the value 0 or 1 to a raster. It reads,

$$
\begin{aligned}
m_l(\omega) &= \prod_{r=1}^{r=k} \omega_{i_r}(t_r) \\
&= \omega_{i_1}(t_1)\omega_{i_2}(t_2)\ldots\omega_{i_k}(t_k) \qquad (2.2)
\end{aligned}
$$

Thus, $m_l(\omega) = 1$ if and only if neuron i_1 fires at time t_1 ... neuron i_k fires at time t_k in the raster ω, and $m_l(\omega) = 0$ otherwise.

Example: The events ensemble $P = \{(0,1),(2,0),(4,2)\}$ corresponds to the monomial $m_l = \omega_0(1)\omega_2(0)\omega_4(2)$.
The monomial's value depends only on the events identified by the set of pairs P (in the example, the set of pairs is: (0,1), (2,0) and (4,2)). The range of the monomial, R, corresponds to the difference between the biggest and the smallest time-stamps in P ($max|t_i - t_j|$). We note $D = R - 1$ (D will be called later on "memory depth"). Hence, a monomial of range R, $m_l(\omega_0^D)$, is a function which associates to the block ω_0^D the values $\{0,1\}$. It is 1 if and only if the events in P are 1 in ω_0^D. It is useful to represent the monomial by a mask:

$$
\begin{pmatrix}
x & x & 1 \\
1 & x & x \\
x & x & x \\
x & 1 & x
\end{pmatrix}
$$

The mask is a block of size $N \times R$ where the entries corresponding to P are set to 1 and the other values are arbitrary (represented by an x). Hence, $m_l = 1$ *if and only if* all the $1s$ in the mask coincide with the $1s$ in the spike block ω_0^{R-1}. The following table shows the value of the monomial m_l in several cases:

If the spike train ω is	0 0 1 1 0 0 0 0 0 0 1 0	0 0 1 0 0 0 1 0 0 1 1 0	0 0 1 1 0 0 0 0 0 0 0 0	1 0 1 1 1 0 0 1 0 0 0 0
then the monomial $m_l = \omega_0(1)\omega_2(0)\omega_3(2) =$	1	0	0	0

There is a difference between "block" and "monomial" that merits to be highlighted. In fact, the pattern represents the activity of all the neurons without any exception. On the opposite, the monomial represents the activity of a precised subset of neurons regardless the activity of other neurons. For

example, the pattern $\begin{smallmatrix}0\\0\\1\end{smallmatrix}$ represent the fact that the neuron 0 fires while the two other neurons are silent. However, the monomial $\begin{smallmatrix}x\\x\\1\end{smallmatrix}$ means that the neuron 0 fires and the rest of the neurons may be silent or active and it takes the value 1 for all the following blocks:

$$\begin{smallmatrix}0\\0\\1\end{smallmatrix}, \begin{smallmatrix}0\\1\\1\end{smallmatrix}, \begin{smallmatrix}1\\0\\1\end{smallmatrix} \text{ and } \begin{smallmatrix}1\\1\\1\end{smallmatrix}$$

2.1.3 Observables

An observable is a function that associates a real value to a spike train. The range of an observable is the smallest integer ($R \geq 1$) such that $f(\omega) = f(\omega_0^{R-1})$. Any observable of range R can be written as a linear combination of monomials with range R.

$$f(\omega_0^{R-1}) = \sum_{l=0}^{2^{NR}-1} f_l m_l(\omega_0^{R-1}) \tag{2.3}$$

A Gibbs potential is such an observable, it reads:

$$\mathcal{H}(\omega) = \sum_{l=0}^{L} \lambda_l m_l(\omega), \tag{2.4}$$

where L is the number of monomials in Gibbs potential and $\lambda_l s$ are free parameters. Although statistical mechanics considers infinite ranges, we restrict here to potentials with finite range R.

2.1.4 Empirical statistics

Empirical statistics (or descriptive statistics) are those which measure quantities directly from data sets. In our case, empirical statistics helps us to visualize information about a spike train ω of length T. Measuring the occurrence of the spike objects (events, spike blocks, spike patterns) we defined in the previous section is an example of empirical statistics we can perform on a spike train.

We mostly use the term "empirical average" which is the occurrence of a spike object over some length. For instance, the empirical firing rate of a neuron i is the empirical average of the event $\omega_i(0)$ (in our notations, neuron i fires) and it reads,

$$\pi_\omega^{(T)}[\,\omega_i\,] = \frac{1}{T}\sum_{t=0}^{T-1} \omega_i(t) \tag{2.5}$$

The empirical average of a spike block of depth D reads:

$$\pi_{\omega}^{(T)}\left[\,\omega_0^D\,\right] = \frac{1}{T-D}\sum_{t=0}^{T-D}\delta(t) \tag{2.6}$$

where

$$\delta(t) = \begin{cases} 1, & \text{if } \omega_t^{t+D} = \omega_0^D \\ 0, & \text{otherwise.} \end{cases}$$

The empirical average of a monomial reads:

$$\pi_{\omega}^{(T)}\left[\,m_l\,\right] = \frac{1}{T-D}\sum_{t=0}^{T-D}\gamma(t) \tag{2.7}$$

where

$$\gamma(t) = \begin{cases} 1, & \text{if } E_{m_l} \subseteq E_{\omega_t^{t+D}} \\ 0, & \text{otherwise.} \end{cases}$$

E_{m_l} and $E_{\omega_t^{t+D}}$ are the ensembles of events that define respectively the monomial m_l and the spike pattern/block ω_t^{t+D}.

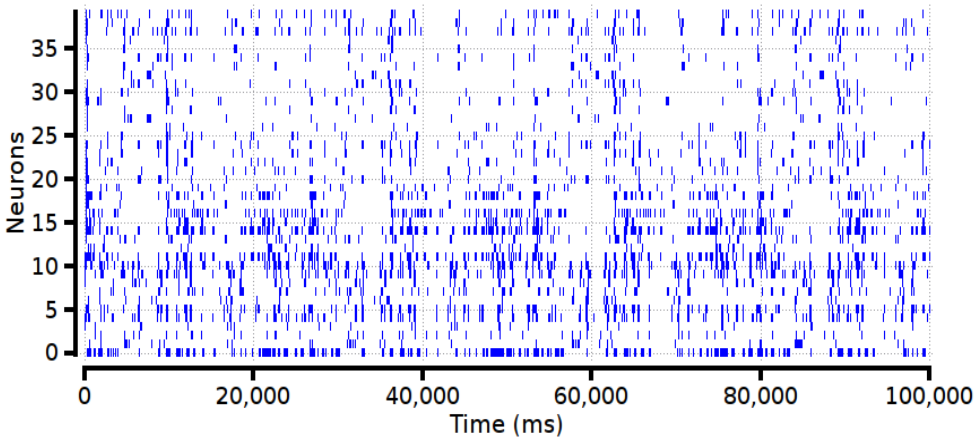
For example, the monomial $\begin{smallmatrix}1\\x\\1\end{smallmatrix}$ whose ensemble of events E_{m_l} is $\{(0,0),(2,0)\}$ belongs to the pattern $\begin{smallmatrix}1\\0\\1\end{smallmatrix}$ and $\begin{smallmatrix}1\\1\\1\end{smallmatrix}$ but does not belong the patterns $\begin{smallmatrix}0\\0\\1\end{smallmatrix}$, $\begin{smallmatrix}0\\0\\0\end{smallmatrix}$ and all the other patterns who have the neurons 0 and 1 silent.

Figure 2.2 shows a real spike train on which we performed some empirical statistics such as the occurrence of single neuron events, patterns and blocks.

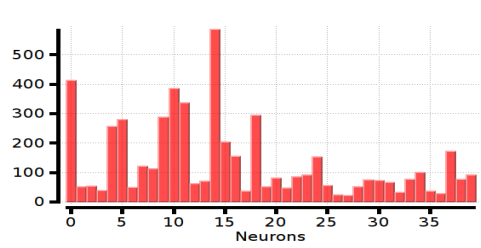
The empirical average is a random variable, depending on the raster ω, as well as on the time length of the sample. It has Gaussian fluctuations whose amplitude tends to 0, as $T \to +\infty$, like $\frac{1}{\sqrt{T}}$ (central limit theorem).

2.1.5 Statistical modeling

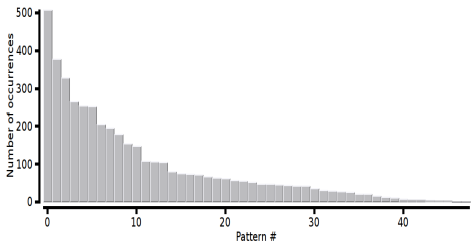
By definition, a model is the simplification of (a complex) reality. Statistical modeling consist in (i) choosing features (in our case monomials) with which we represent the statistics of random variables (in our case spike blocks) in a process or a system (in our case spike train) and (ii) finding the best distribution that corresponds to the data. The monomials choice affects the goodness of the model. For instance, if we choose a Bernoulli model (where all neurons are independent) to represent the statistics of patterns in a spike train where instantaneous activity between neurons is significant, then the model is not good. We evaluate the goodness of a model using two techniques



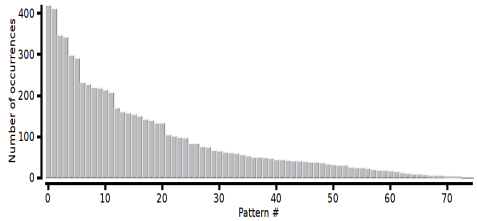
(a) A real spike train of 40 neurons)



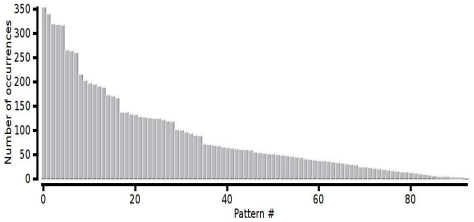
(b) The counts of single neurons activities



(c) The histogram of pattern of depth 1



(d) The histogram of pattern of depth 2



(e) The histogram of pattern of depth 3

Figure 2.2: An example of the empirical statistics measured on a real spike train of 40 neurons (a). (b) the counts of single neurons activities. (c), (d), and (e) show respectively the histograms of patterns of depth 1, 2 and 3.

explained below (the Kullback-Leibler divergence (KLD) and the confidence plots).

Our aim is to find a statistical distribution μ that is the closest possible to the empirical distribution π. We will explain in the next section how we infer μ from π.

Kullback-Leibler divergence

The KLD between two probability distribution reads:

$$d_{kl}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}}) = \lim_{n\to+\infty} \frac{1}{n} \sum_{\omega_0^n} \pi_\omega^{(T)}[\,\omega_0^n\,] \log\left(\frac{\pi_\omega^{(T)}[\,\omega_0^n\,]}{\mu_{\boldsymbol{\lambda}}[\omega_0^n]}\right) \tag{2.8}$$

In the spatial case, the KLD reads:

$$d_{kl}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}}) = \sum_{\omega(0)} \pi_\omega^{(T)}[\,\omega(0)\,] \log\left(\frac{\pi_\omega^{(T)}[\,\omega(0)\,]}{\mu_{\boldsymbol{\lambda}}[\omega(0)]}\right) \tag{2.9}$$

Computing the KLD gives an idea about the distance between the model and the spike train. Theoretically, the smaller the KLD, the better the model.

Confidence plots

Confidence plots is another way to evaluate a model. Since we are expecting an error on the predicted probabilities of blocks, an interval of error should be fixed in order to check whether the estimated values lie within this interval or not. The expected predicted probabilities of monomials are random variables whose exact value is equal to the expected value $\pm$ some error. The relation between "expected" and "true" value is given by the central limit theorem which states the expected averages oscillates around the true value as a normal distribution of mean equal to the true value of the monomial average and a standard deviation equal to $\sigma = \sqrt{\frac{\pi(\omega_0^n)\times(1-\pi(\omega_0^n))}{T}}$. Figure 2.3 show a Gaussian distribution and the $68-95-99.7$ rule. The region of $\pi(\omega_0^n) \pm \sigma$ covers 68% of the accepted expected averages and so on. We have chosen to choose a confidence bounds which ensures that we cover 99.7% of the expected averages. For that, our confidence bounds read:

$$C_p = \pi(\omega_0^n) \pm 3\sqrt{\frac{\pi(\omega_0^n) \times (1-\pi(\omega_0^n))}{T}}. \tag{2.10}$$

An example of confidence plots will be shown in following chapters.

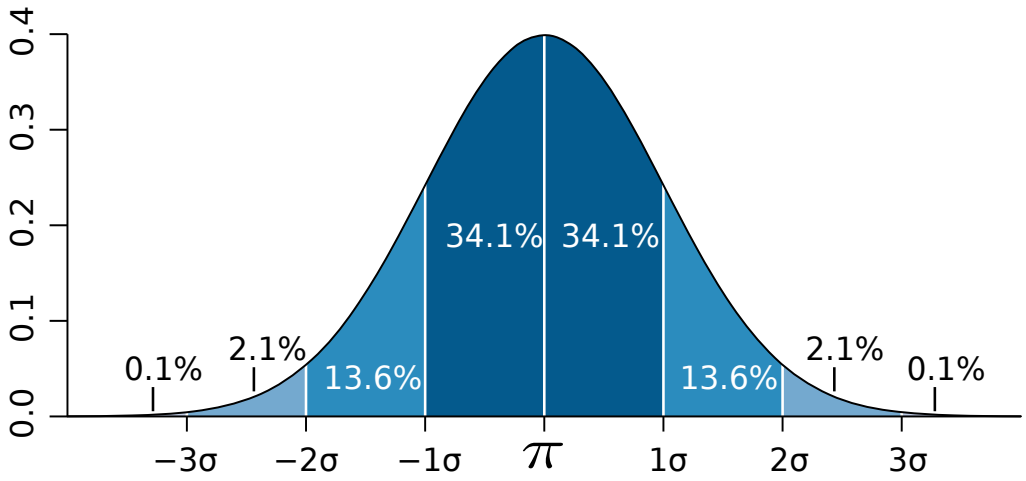


Figure 2.3: The $68-95-99.7$ rule. It states that 68% of the values lie within $[\pi-\sigma, \pi+\sigma]$, 95% of the values lie within $[\pi-2\sigma, \pi+2\sigma]$ and 97.3% of the values lie within $[\pi-3\sigma, \pi+3\sigma]$.

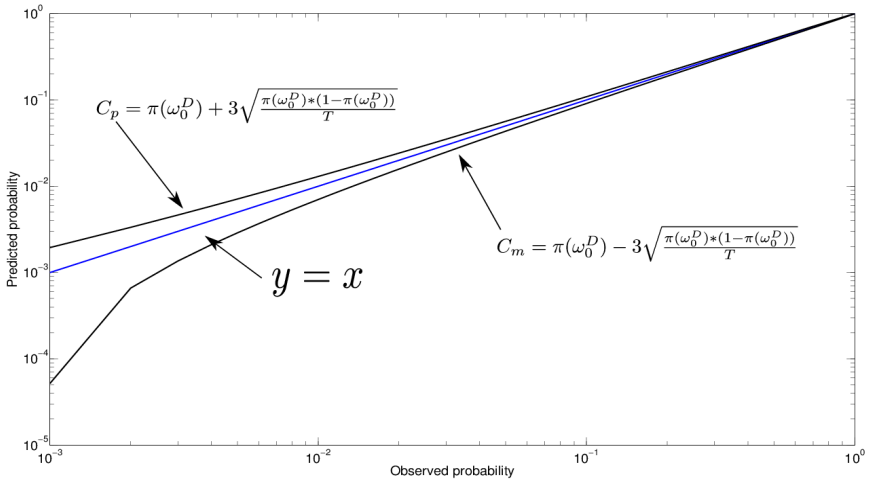


Figure 2.4: The confidence bounds, C_m and C_p around the equality identity $y = x$.

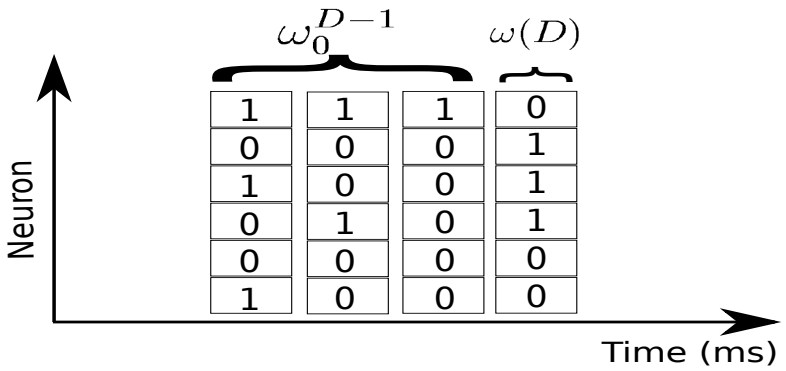


Figure 2.5: An illustration of transition probability presenting the block ω_0^{D-1} (for illustration, here $D = 3$) and the pattern $\omega(D)$.

2.1.6 Markov Chain

In this thesis, we consider that the observed spike trains have been generated by a Markov chain with memory depth D, defined by a family of transition probabilities $P[\omega(D)|\omega_0^{D-1}]$ (see Figure 2.5 for an illustration). This consideration results from the temporal dependencies assumption. In general, those transition probabilities depend explicitly on time, but we focus here on time-translation invariant processes (homogeneous Markov chain). As a consequence, the Markov chain is fully described by the probabilities $P[\omega(D)|\omega_0^{D-1}]$ (in the general case, one would have to consider transition probabilities $P_n[\omega(n)|\omega_{n-D}^{n-1}]$, where the probability depends explicitly on the time index n). A probability distribution $\mu(\omega_0^{D-1})$ is compatible with the Markov chain if, for any block ω_0^D:

$$\mu[\omega_1^D] = \sum_{\omega(0)} P[\omega(D)|\omega_0^{D-1}]\mu(\omega_0^{D-1}) \tag{2.11}$$

μ is also called the invariant probability of the Markov chain. We shall assume here that all transition probabilities are strictly positive, $P[\omega(D)|\omega_0^{D-1}] > 0$. In this case, the invariant probability exists and is unique. Given the set of transition probabilities and the invariant probability μ, the probability of blocks of depth $n + 1$ is given by the Chapman-Kolmogorov equation:

$$\begin{aligned}\mu[\omega_0^n] &= \mu[\omega_0(n), \omega_0^{n-1}] \\ &= \mu[\omega(n)|\omega_0^{n-1}]\mu(\omega_0^{n-1}), \\ &= \mu[\omega(n)|\omega_0^{n-1}]\mu[\omega(n-1)|\omega_0^{n-2}]\mu(\omega_0^{n-2}) \\ &= \prod_{r=D}^{n} P[\omega(r)|\omega_{r-D}^{r-1}])\mu[\omega_0^{D-1}] \end{aligned} \tag{2.12}$$

Eq.(2.12) states that the probability to observe the block ω_0^n depends on the $\mu-$probability to start with the initial state ω_0^{D-1} as well as on the transition probabilities, taking into account, each time, how likely it is to obtain a pattern $\omega(r)$ given the previous state ω_{r-D}^{r-1}.

On the opposite, if $D = 0$, the probability to have the spike pattern $\omega(D)$ does not depend on the past activity of the network (we call this the memory-less case). In this case $P[\omega(r)|\omega_{r-D}^{r-1}] = \mu[\omega(r)]$ and the probability (Eq.2.12) of a block becomes:

$$\mu[\omega_0^n] = \prod_{l=0}^{n} \mu[\omega(l)] \tag{2.13}$$

Therefore, in the memory less case, spikes occurring at different times are independent. This emphasizes the deep differences between the case $D = 0$ and $D > 0$.

2.2 Maximum Entropy Principle

The Maximum Entropy Principle (MEP) is a method to obtain, from the observation of a statistical sample, a probability distribution that approaches at best the statistics of the sample, taking only into account the information available on this sample. In general, it relies on a fundamental assumption: the statistical model that has generated the sample is *stationary*, although some attempts to extend it to non stationary data has been proposed in [Pressé et al., 2013]. This means that the average of an observable does not depend explicitly on time.

Assigning equal probabilities (uniform probability distribution) to possible outcomes dates back to Laplace and Bernoulli ([Garibaldi and Penco, 1985]) ("Principle of insufficient reason"). Maximizing the statistical entropy *without* constraints is equivalent to this principle. When one has some knowledge about the outcomes, this knowledge has to be used to infer the probabilities of these outcomes. In general knowledge is characterized by empirical average

of prescribed observables: this constitutes a set of constraints. Maximizing the statistical entropy given those constraints provides a distribution as far as possible from the uniform and as close as possible to the empirical distribution. For instance, considering the empirical mean and variance of the sample of a random variable as constraints results in a Gaussian distribution as a prototype to fit the statistics of this sample. In our case, the natural constraints are represented by the probability of occurrence of characteristic spike events in the spike train, or, equivalently, by the average of specific monomials. Classical examples of constraints are the probability that a neuron fires at a given time (firing rate) or the probability that two neurons fire at the same time.

2.2.1 Motivations

The Maximum Entropy Method provides a method to infer a Markov chain of depth D, that fits best to data. In this thesis, D is fixed by the user, i.e., we haven not investigated a method to guess D from data (as proposed in [Galves et al., 2012] and [Csiszár and Talata, 2006]). The goal is to find a probability distribution μ such that:

- μ is inferred from an empirical raster ω, by computing the empirical average of a set of predefined monomials of range $R = D+1$. One asks that the average of m_l with respect to μ satisfies:

$$\mu\left[\, m_l \,\right] = \pi_\omega^{(T)}\left[\, m_l \,\right], \quad l = 1, \ldots, L. \tag{2.14}$$

  The mean of m_l, predicted by μ is equal to the mean computed on the experimental raster. The set of monomials m_l defines the model μ.

- μ has to be "as simple as possible", with the least structure and a minimum number of tunable parameters. In the Maximum Entropy paradigm ([Jaynes, 1957]) these issues are (partly) solved by seeking a probability distribution μ which maximizes the entropy under the constraints (Eq.2.14). The entropy is defined explicitly below (see Eq. 2.15 and 2.35) in our context.

- From the knowledge of μ one can compute the probability of arbitrary blocks (via Eq.2.12) and the average of other monomials than the m_ls.

In the following, we will present the mathematical framework for inferring Maximum Entropy models from spike trains. The simplest form of the Maximum Entropy Principle is to consider all neurons are independent. As

we add more monomial, for instance those which represents interactions between neurons, the model and the computation get more complicated. In addition, adding temporal effects is not trivial and changes completely the mathematical framework. For that, we will begin by explaining the simplest forms of Maximum Entropy, such as Bernoulli and Ising where monomials are only spatial. Afterwards, we present the one time step Maximum Entropy models and finally the spatio-temporal Maximum Entropy models which we use as a general framework for both spatial and spatio-temporal models.

2.2.2 Maximum Entropy for spatial models

Spatial models have been studied widely in the literature, [Ohiorhenuan et al., 2010], [Schneidman et al., 2006], [Tkačik et al., 2009], [Ganmor et al., 2011a]. For instance, [Schneidman et al., 2006] aimed at unraveling the role of instantaneous pairwise correlations in retina spike trains. Although these correlations are weak, these researchers investigated whether they play a more significant role in spike train statistics than firing rates. For instance, such models would contain monomials that correspond to rates and $n-uplets$ instantaneous interactions such as pairwise, triplets .... Firing rates correspond to the average of monomials of the form $\omega_i(0), i = 0, \ldots, N-1$ (the time index 0 comes from the assumed time-translation invariance) while instantaneous pairwise correlations correspond to averages of observables of the form $\omega_i(0)\omega_j(0), 0 \leq i < j \leq N-1$. Here, in the spatial case, we are assuming that the spike train generation process has no memory. The entropy of μ is:

$$\mathcal{S}[\,\mu\,] = -\sum_{\omega(0)} \mu[\,\omega(0)\,] \log \mu[\,\omega(0)\,], \tag{2.15}$$

where the sum holds on the 2^N possible spike patterns $\omega(0)$. Note that the time index (here 0) does not play a role since we have assumed μ to be stationary.

The Maximum Entropy problem is now stated as follows. Find *the* distribution μ that maximizes the entropy (Eq.2.15) given the constraints (Eq.2.14). There is, additionally, the probability normalization constraint:

$$\sum_{\omega(0)} \mu[\,\omega(0)\,] = 1. \tag{2.16}$$

This provides a variational problem

$$\mu = \arg\max_{\nu \in \mathcal{M}} \left[\mathcal{S}\left[\,\nu\,\right] + \lambda_0 \left(\sum_{\omega(0)} \nu\left[\,\omega(0)\,\right] - 1 \right) + \sum_{l=1}^{L} \lambda_l \left(\nu\left[\,m_l\,\right] - \pi_\omega^{(T)}\left[\,m_l\,\right] \right) \right], \tag{2.17}$$

where $\mathcal{M}$ is the set of (stationary) probabilities on rasters.

Stated in this form, the Maximum Entropy is a Lagrange multipliers problem. The sought probability distribution is the classical Gibbs distribution:

$$\mu_{\boldsymbol{\lambda}}\left[\,\omega(0)\,\right] = \frac{1}{Z_{\boldsymbol{\lambda}}} e^{\mathcal{H}[\,\omega\,]}, \tag{2.18}$$

where

$$Z_{\boldsymbol{\lambda}} = \sum_{\omega(0)} e^{\mathcal{H}[\,\omega\,]}, \tag{2.19}$$

is the *partition function*, and $\mathcal{H} = \sum_{l=1}^{L} \lambda_l m_l$. Note that, in general the number of monomials considered in the practical applications of the Maximum Entropy is much smaller than their possible number $L(N,R) = 2^{NR}$. Thus, here, most of the coefficients in Figure 2.4 are constrained to zero. The value of the non zero λ_ls is fixed by the relation:

$$\mu\left[\,m_l\,\right] = \frac{\partial \log Z_{\boldsymbol{\lambda}}}{\partial \lambda_l} = \pi_\omega^{(T)}\left[\,m_l\,\right],\ l = 1 \ldots L. \tag{2.20}$$

Additionally, note that the matrix $\frac{\partial^2 \log Z_{\boldsymbol{\lambda}}}{\partial \lambda_l \, \partial \lambda_{l'}}$ is positive. This ensures the convexity of the problem and the uniqueness of the solution.

Note that $\log Z_{\boldsymbol{\lambda}}$ does not only allow us to obtain the averages of the observables, it also allows to characterize fluctuations. If a raster is distributed according to the Gibbs distribution (Eq.2.18), then, as pointed out in section 2.1.4, the empirical average of an observable has fluctuations. One can show that these fluctuations are Gaussian (Central limit theorem). The joint probability of $\pi_\omega^{(T)}\left[\,m_l\,\right]$, $l = 1, \ldots, L$ is Gaussian, with mean $\mu\left[\,m_l\,\right]$ given by Eq. 2.20 and covariance matrix $\frac{\Sigma}{T}$ where the matrix Σ has entries:

$$\Sigma_{jk} = \frac{\partial^2 \log Z_{\boldsymbol{\lambda}}}{\partial \lambda_j \partial \lambda_k}. \tag{2.21}$$

In general, we do not expect that μ to be equal to the (hidden) probability shaping the observed sample. It is only the closest one satisfying the constraints (Eq. 2.14). The notion of closeness is related to the Kullback-Leibler divergence, defined in the next section.

Let us now discuss what this principle gives in two spatial cases considered:

(i) **Only firing rates are constrained.** Then:

$$\mathcal{H}(\omega(0)) = \sum_{i=0}^{N-1} \lambda_i \omega_i(0). \tag{2.22}$$

It can be shown that the corresponding probability μ is:

$$\mu\,[\,\omega(0)\,] = \prod_{i=0}^{N-1} \frac{e^{\lambda_i\,\omega_i(0)}}{1+e^{\lambda_i}}.$$

Thus, the corresponding statistical model is such that spikes emitted by distinct neurons at the same time are independent. The parameter λ_i is directly related to the firing rate r_i since $r_i = \mu\,[\,\omega_i(0) = 1\,] = \frac{e^{\lambda_i}}{1+e^{\lambda_i}}$, so that we have:

$$\mu\,[\,\omega_0^n\,] = \prod_{l=0}^{n} \prod_{i=0}^{N-1} r_i^{\omega_i(l)}\,(1-r_i)^{1-\omega_i(l)},$$

the classical probability of coin tossing with independent probabilities (*Bernoulli model*). Thus, fixing only the rates as constraints, the Maximum Entropy principle leads to analyze spike statistics as if each spike were thrown randomly and independently, as with coin tossing. This is the "most random model", which has the advantage of making as few hypothesis as possible. However, when only constrained with mean firing rates, the prediction of even small spike blocks in the retina was not successful. This was expected since this model assumes independence between neurons, an assumption that has been proven wrong in earlier studies (e.g. [Puchalla et al., 2005]).

(ii) **Firing rates and pairwise correlations are constrained.** In the second model (introduced e.g. by [Schneidman et al., 2006]), The Maximum Entropy model is constrained with both mean firing rates and instantaneous pairwise correlations between neurons. In this case,

$$\mathcal{H}(\omega(0)) = \sum_{i=0}^{N-1} \lambda_i\,\omega_i(0) + \sum_{i,j=0}^{N-1} \lambda_{ij}\,\omega_i(0)\,\omega_j(0). \tag{2.23}$$

Here the potential can be identified with the Hamiltonian of a magnetic system with binary spins. It is thus often called "Ising model" in the spike train analysis literature, although the original Ising model has constant and positive couplings, [Georgii, 1988]. The corresponding statistical model is the least structured model respecting these first and second order pairwise instantaneous constraints. The number of parameters is of the order[1] of N^2, to be compared with the 2^N possible patterns.

Schneidman et al showed that the Ising model model successfully predicts spatial patterns, a result which was confirmed by [Shlens et al., 2006] (see [Nirenberg and Victor, 2007] for a review). Other works have used the same method and found also a good prediction in cortical structure in-vitro, [Tang et al., 2008], and in the visual cortex in vivo, [Yu et al., 2008]. Later on, several authors considered higher order terms still corresponding to $D = 0$ ([Ohiorhenuan et al., 2010], [Schneidman et al., 2006], [Tkačik et al., 2009], [Ganmor et al., 2011]). Note that these results have been obtained on relatively small subsets of neurons (usually groups of 10). An interesting challenge is to test how these results hold for larger subsets of neurons, and if other constraints have to be added, [Ganmor et al., 2011b].

2.2.3 One time step spatio-temporal models

From the statistical mechanics point of view, a natural extension of the previous formalism consists of considering the space of rasters Ω as a lattice where one dimension is "space" (neurons index) and the other is time. The idea is then to consider a potential still of the form (2.4) but where the observables correspond to spatio-temporal events. We assume that $\mathcal{H}$ has range $R = D + 1$, $0 \leq D < +\infty$. The potential of a spike block ω_0^n, $n \geq D$ is:

$$\mathcal{H}\left(\omega_0^n\right) = \sum_{l=0}^{n-D} \mathcal{H}\left(\omega_l^{D+l}\right) \tag{2.24}$$

On this basis, restricting to the case where $D = 1$ (one time step memory depth) Marre et al have proposed in [Marre et al., 2009] to construct a Markov chain, where transition probabilities $P\left[\omega(l+1) \,|\, \omega(l)\right]$ are proportional to $e^{\mathcal{H}(\omega_l^{l+1})}$. If μ is the invariant probability of that chain, the application of Eq. (2.12) leads to probability of blocks $\mu\left[\omega_0^n\right]$, proportional to $e^{\mathcal{H}\left(\omega_0^n\right)}$: the probability of a block is proportional to the exponential of

[1] Most approaches assume moreover that the pairwise coefficients are symmetric $\lambda_{kl} = \lambda_{lk}$ which divides the number of parameters by 2.

its potential ("energy"). This approach is therefore quite natural from the statistical mechanics point of view.

The main problem, however, is "what is the proportionality coefficient ?" As shown in [Marre et al., 2009], the normalization of conditional probabilities does not reduce to the mere division by a constant partition function. This normalization factor is itself dependent on the past activity.

To overcome this dependency, Marre et al assumed that the activity respected a detailed balance. In this particular case, it can be shown that the normalization factor becomes, again, a constant. But this is an important reduction that could have implications for the interpretation of the data: for example, with this simplification, it is not possible to give an account of asymmetric cross-correlograms.

2.2.4 General spatio-temporal models

We now present the general formalism which allows to solve the variational problem "maximizing entropy under spatio-temporal constraints". This approach is rigorous and the normalization problem is resolved without requiring additional assumptions such as detailed balance. At the end of this section, we briefly discuss other approaches considering spatio-temporal statistics and their relations to potentials of the form (2.4).

Markov Chain for spatio-temporal models

In this section we show how one can generate a Markov chain where transition probabilities are proportional to $e^{\mathcal{H}(\omega_l^{l+D})}$, for a potential $\mathcal{H}$ corresponding to spatio-temporal events. We also solve the normalization problem. This construction is well known and is based on the so-called transfer matrix (see e.g. [Georgii, 1988] for a presentation in the context of statistical physics; [Parry and Pollicott, 1990] for a presentation in the context of ergodic theory and [Vasquez et al., 2012] for a presentation in the context of spike trains analysis).

This matrix is constructed as follows. Consider two spike blocks w_1, w_2 of range $D \geq 1$. The transition $w_1 \to w_2$ is *legal* if w_1 has the form $\omega(0)\omega_1^{D-1}$ and w_2 has the form $\omega_1^{D-1}\omega(D)$. The vectors $\omega(0), \omega(D)$ are arbitrary but the block ω_1^{D-1} is common. Here is an example of a legal transition :

$$w_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}; \; w_2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Here is an example of a forbidden transition

$$w_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}; \; w_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Any block ω_0^D of range $R = D + 1$ can be viewed as a legal transition from the block $w_1 = \omega_0^{D-1}$ to the block $w_2 = \omega_1^D$ and in this case we write $\omega_0^D \sim w_1 w_2$.

The *transfer matrix* $\mathcal{L}$ is defined as:

$$\mathcal{L}_{w_1,w_2} = \begin{cases} e^{\mathcal{H}(\omega_0^D)} & \text{if} \quad w_1, w_2 \quad \text{is legal with } \omega_0^D \sim w_1 w_2 \\ 0, & \text{otherwise.} \end{cases} . \tag{2.25}$$

From the matrix $\mathcal{L}$ the transition matrix of a Markov chain can be constructed, as we now show. Since observables are usually assumed to be bounded from below, $\mathcal{H}(\omega_0^D) > -\infty$, thus $e^{\mathcal{H}(\omega_0^D)} > 0$ for each legal transition. As a consequence of the Perron-Frobenius theorem [Gantmacher, 1998, Seneta, 2006], $\mathcal{L}$ has a unique real positive eigenvalue $s_{\boldsymbol{\lambda}}$, strictly larger than the modulus of the other eigenvalues (with a positive gap), and with right, R, and left, L, eigenvectors: $\mathcal{L}R = s_{\boldsymbol{\lambda}} R$, $L\mathcal{L} = s_{\boldsymbol{\lambda}} L$, or, equivalently[2]:

$$\sum_{\omega(D)\in\{0,1\}^N} e^{\mathcal{H}\left(\omega_0^D\right)} R\left(\omega_1^D\right) = s_{\boldsymbol{\lambda}} R\left(\omega_0^{D-1}\right);$$

$$\sum_{\omega(0)\in\{0,1\}^N} L\left(\omega_0^{D-1}\right) e^{\mathcal{H}\left(\omega_0^D\right)} = s_{\boldsymbol{\lambda}} L\left(\omega_1^D\right).$$

These eigenvectors have strictly positive entries $R(\,.\,) > 0$, $L(\,.\,) > 0$, functions of blocks of range D. They can be chosen so that the scalar product $\langle L, R| = \rangle 1$. We define:

$$\mathcal{P}(\mathcal{H}) = \log s_{\boldsymbol{\lambda}}, \tag{2.26}$$

called "topological pressure or free energy". We discuss the origin of this term and its properties below.

To define a Markov chain from the transfer matrix $\mathcal{L}$ (Eq. 2.25) we introduce the *normalized potential*:

$$\phi(\omega_0^D) = \mathcal{H}(\omega_0^D) - \mathcal{G}_{\boldsymbol{\lambda}}(\omega_0^D) \tag{2.27}$$

[2]The right eigenvector R has 2^{ND} entries R_w corresponding to blocks of range D. It obeys $\sum_{w_2} \mathcal{L}_{w_1 w_2} R_{w_2} = s_{\boldsymbol{\beta}} R_{w_1}$, where $w_1 = \omega_0^{D-1}$ and where the sum runs over blocks $w_2 = \omega_1^D$. Since $\mathcal{L}_{w_1 w_2}$ is non zero only if the entries w_1, w_2 have the block ω_1^{D-1} in common, and since the right hand side $(s_{\boldsymbol{\beta}} R_{w_1})$ fixes the value of w_1, this sum holds in fact on all possible values of $\omega(D)$. The notation R_w, although natural, does not make explicit the block involved. This is problematic when one wants to handle equations such as Eq. 2.32. As a consequence, we prefer to use the notation $R(\,block\,)$ to make explicit this dependence. The same remark holds mutatis mutandis for the left eigenvector.

with:

$$\mathcal{G}_{\boldsymbol{\lambda}}(\omega_0^D) = \log R\left(\omega_0^{D-1}\right) - \log R\left(\omega_1^D\right) + \log s_{\boldsymbol{\lambda}}, \tag{2.28}$$

and a family of transition probabilities:

$$P\left[\omega(D) \,\middle|\, \omega_0^{D-1}\right] \stackrel{\text{def}}{=} e^{\phi(\omega_0^D)} > 0. \tag{2.29}$$

These transition probabilities define a Markov chain which admits a unique invariant probability:

$$\mu(\omega_0^{D-1}) = R\left(\omega_0^{D-1}\right) L\left(\omega_0^{D-1}\right). \tag{2.30}$$

From the general form of block probabilities (Eq. 2.12) the probability of blocks of depth $n \geq D$ is, in this case :

$$\mu\left[\omega_0^n\right] = e^{\sum_{l=0}^{n-D} \phi\left(\omega_l^{D+l}\right)} \mu\left[\omega_0^{D-1}\right]. \tag{2.31}$$

thus, from Eq. 2.30, 2.27 and 2.28:

$$\mu\left[\omega_0^n\right] = \frac{e^{\mathcal{H}\left(\omega_0^n\right)}}{s_{\boldsymbol{\lambda}}^{n-D+1}} R\left(\omega_{n-D+1}^n\right) L\left(\omega_0^{D-1}\right), \tag{2.32}$$

where $\mathcal{H}\left(\omega_0^n\right)$ is given by Eq. 2.24.

Remarks

1. We have been able to compute the probability of any blocks ω_0^n. It is proportional to $e^{\mathcal{H}\left(\omega_0^n\right)}$ and the proportionality factor has been computed. In the general case of spatio-temporal events, it depends on ω_0^{D-1} and ω_{n-D+1}^n.

   The same arises in statistical mechanics when dealing with boundary conditions. The forms (2.31) and (2.32), remind Gibbs distributions on spin lattices, with lattice translation-invariant probability distributions given specific boundary conditions. Given a spin-potential of spatial-range n, the probability of a spin block depends upon the state of the spin block, as well as spins states in a neighborhood of that block. The conditional probability of this block given a fixed neighborhood is the exponential of the energy characterizing physical interactions, within the block, as well as interactions with the boundaries. In Eq. 2.31, spins are replaced by spiking patterns; space is replaced by time. Spatial boundary conditions are here replaced by the dependence upon ω_0^{D-1} and ω_{n-D+1}^n.

As a consequence, as soon as one is dealing with spatio-temporal events the normalization of conditional probabilities does not reduce to the mere division by:

$$Z_n = \sum_{\omega_0^n} e^{\mathcal{H}\left(\omega_0^n\right)}, \tag{2.33}$$

as easily checked in Eq. (2.32).

2. The topological pressure obeys nevertheless:

$$\mathcal{P}(\mathcal{H}) = \lim_{n \to +\infty} \frac{1}{n} \log Z_n, \tag{2.34}$$

and is analogue to a thermodynamic potential density (free energy, free enthalpy, pressure). This analogy is also clear in the variational principle (Eq. 2.37) below. To our best knowledge the term "topological pressure" has its roots in the thermodynamic formalism of hyperbolic (chaotic) maps [Ruelle, 1978, Parry and Pollicott, 1990, Beck and Schloegl, 1995]. In this context, this function can be computed as the grand potential of the grand canonical ensemble, as a cycle expansion over unstable periodic orbits. It is therefore equivalent to a pressure[3] depending on topological properties (periodic orbits).

3. In the case $D = 0$ the Gibbs distribution reduces to (Eq. 2.18). One can indeed easily show that:

$$\exp \mathcal{G}_{\boldsymbol{\lambda}} = s_{\boldsymbol{\lambda}} = \sum_{\omega(0)} e^{\mathcal{H}(\omega(0))} = Z_{\boldsymbol{\lambda}},$$

where $Z_{\boldsymbol{\lambda}}$ is the partition function (Eq. 2.19). Additionally, since spike patterns occurring at distinct time are independent in the $D = 0$ case, Z_n in Eq. 2.33 can be written as $Z_n = Z_{\boldsymbol{\lambda}}^n$ so that $\mathcal{P}(\mathcal{H}) = \log Z_{\boldsymbol{\lambda}}$.

4. In the general case of spatio-temporal constraints, the normalization requires the consideration of normalizing function $\mathcal{G}_{\boldsymbol{\lambda}}$ *depending as well on the blocks* ω_0^D. Thus, in addition to function $\mathcal{H}$ normalization introduces a second *function* of spike blocks. This increases consequently the complexity of Gibbs potentials and Gibbs distributions compared to the spatial ($D = 0$) case where $\mathcal{G}_{\boldsymbol{\lambda}}$ reduces to a constant.

[3] The grand potential Φ obeys $\Phi = -PV$, where P is the physical pressure and V the volume. Therefore, the grand potential density is (minus) the physical pressure.

2.2.5 Application of the Maximum Entropy principle

We now show that the probability distribution defined this way solves the variational problem "maximizing entropy under spatio-temporal constraints".

We define the *entropy rate* (or Kolmogorov-Sinai entropy):

$$S\left[\,\mu\,\right] \;=\; -\,\limsup_{n\to\infty}\frac{1}{n+1}\sum_{\omega_0^n}\mu\left[\,\omega_0^n\,\right]\log\mu\left[\,\omega_0^n\,\right], \tag{2.35}$$

where the sum holds over all possible blocks ω_0^n. Note, that in the case of a Markov chain $S\left[\,\mu\,\right]$ also reads [Cornfeld et al., 1982]:

$$S\left[\,\mu\,\right] \;=\; -\sum_{\omega_0^D}\mu\left[\,\omega_0^D\,\right]\,P\left[\,\omega(D)\,\middle|\,\omega_0^{D-1}\,\right]\log P\left[\,\omega(D)\,\middle|\,\omega_0^{D-1}\,\right], \tag{2.36}$$

whereas, when $D=0$, $s\left[\,\mu\,\right]$ reduces to the definition (Eq. 2.15).

As a general result from ergodic theory [Ruelle, 1978, Keller, 1998, Chazottes and Keller, 2008] and mathematical statistical physics [Georgii, 1988], there is a unique[4] probability distribution μ such that [Ruelle, 1978, Keller, 1998, Chazottes and Keller, 2008]:

$$\mathcal{P}\left[\,\mathcal{H}\,\right] = \sup_{\nu\in\mathcal{M}_{inv}}\left(\,h\left[\,\nu\,\right] + \nu\left[\,\mathcal{H}\right)\,]\,\right) = S\left[\,\mu\,\right] + \mu\left[\,\mathcal{H}\,\right], \tag{2.37}$$

where $\mathcal{P}\left[\,\mathcal{H}\,\right]$ is given by Eq. 2.26. $\mathcal{M}_{inv}$ is the set of all possible time-translation invariant probabilities on the set of rasters with N neurons and $\nu\left[\,\mathcal{H}\,\right] = \sum_{\omega_0^D}\mathcal{H}(\omega_0^D)\,\nu(\omega_0^D)$ is the average value of $\mathcal{H}$ with respect to the probability ν.

Looking at the second equality, the variational principle (Eq. 2.37) selects, among all possible probabilities ν, a unique one realizing the supremum. This is exactly the invariant distribution of the Markov chain and is the sought Gibbs distribution. It is clear from Eq. 2.37 that the topological pressure is the formal analogue to a thermodynamic potential density, where $\mathcal{H}$ somewhat fixes the "ensemble": $\nu\left[\,\mathcal{H}\,\right] = \sum_{k=1}^{K}\lambda_l\nu\left[\,m_l\,\right]$ plays the role of λE (canonical ensemble), $\lambda E - \mu N$ (grand canonical ensemble), ... in thermodynamics [Beck and Schloegl, 1995].

[4] The result is straightforward here since we consider bounded potentials with finite range.

2.3 Inferring the parameters λ_l

The inverse problem of finding the values of λ_ls from the observables average measured on the data is a hard problem with no exact analytical solution. However, in the context of spatial models with pairwise interactions the wisdom, coming from statistical physics and especially Ising model and spin-glasses, as well as from the Boltzmann machine learning community, can be used. As a consequence, in this context, several strategies were proposed. Ackley et al [Ackley et al., 1985] proposed a technique to estimate the parameters of a Boltzmann machine. This technique is effective for small networks but it is time consuming. In practice, the time necessary to learn the parameters increases exponentially with the number of units. To speed up the parameters estimation, analytical approximations of the inverse problem have been proposed, which express the parameters λ_l as a nonlinear function of the correlations of the activity (see for example [Tanaka, 1998], [Roudi et al., 2009a], [Sessak and Monasson, 2009], [Ackley et al., 1987], [Roudi et al., 2009b], [Ackley et al., 1985], [Higuchi and Mezard, 2009], [Kappen and Rodriguez, 1998]).

These methods do not give an exact result, but are computationally fast. We do not pretend to review all of them here, but we quote a few prominent examples. In [Sessak and Monasson, 2009], the authors proposed a systematic small-correlation expansion to solve the inverse Ising problem. They were able to compute couplings up to the third order in the correlations for generic magnetizations, and to the seventh order in the case of zero magnetizations. Their resulting expansion outperforms existing algorithms on the Sherrington Kirkpatrick spin-glass model [Sherrington and Kirkpatrick, 1975].

Based on a high-field expansion of the Ising thermodynamic potential, Cocco et al [Cocco et al., 2009] designed an algorithm to calculate the parameters in a time polynomial with N, where the couplings are expressed as a weighted sum over the power of the correlations. They did not obtained a closed analytical expression, but their algorithm could run in a time that was polynomial in the number of neurons.

Other methods, based on Thouless-Anderson-Palmer equations [Thouless et al., 1977] and linear response [Kappen and Rodriguez, 1998], or information geometry [Tanaka, 1998], initially proposed in the field of spin-glasses, have been adapted and applied to spike train analysis (see e.g. the work done by Roudi and collaborators [Roudi and Hertz, 2011]).

The success of these approximations depends on the data set, and there is no a priori guarantee about their efficiency at finding the right values of the parameters. However, by getting closer to the correct solution, they can potentially speed up the convergence of the learning by starting with a seed

much closer to the real solution than if taking a random starting point.

Note also that all the techniques mentioned above have been designed for the case where there is no temporal interaction (except [Cocco et al., 2009, Roudi and Hertz, 2011] which are discussed in the section 1.4). Now, we explain how the parameters estimation can be done in the spatio-temporal models.

In the general case parameters λ_ls can be determined thanks to the following properties.

- $\mathcal{P}[\mathcal{H}]$ is a log generating function of cumulants. First:

$$\frac{\partial \mathcal{P}[\mathcal{H}]}{\partial \lambda_l} = \mu[m_l]. \tag{2.38}$$

  This is an extension of Eq. 2.20 to the time-dependent case.

- Second:

$$\frac{\partial^2 \mathcal{P}[\mathcal{H}]}{\partial \lambda_l \partial \lambda_k} = \frac{\partial \mu[m_l]}{\partial \lambda_k} = \sum_{n=-\infty}^{+\infty} C_{m_l m_k}(n), \tag{2.39}$$

  where $C_{m_l\, m_k}(n)$ is the correlation function between the two monomials m_l and m_k at time n. Note that correlation functions decay exponentially fast whenever $\mathcal{H}$ has finite range. So that $\sum_{n=-\infty}^{+\infty} C_{m_l\, \mathbf{m}_l}(n) < +\infty$.

  Eq. 2.39 characterizes the variation in the average value of m_l when varying λ_l (linear response). The corresponding matrix is a susceptibility matrix. It controls the Gaussian fluctuations of observables around their mean (central limit theorem) [Ruelle, 1978, Parry and Pollicott, 1990, Chazottes and Keller, 2008]. This is the generalization of Eq. 2.21 to the time dependent case. As a particular case, the fluctuations of the empirical average $\pi_\omega^{(T)}[m_l]\,\omega$ of m_l around its mean $\mu[m_l]$ are Gaussian with a mean-square deviation $\frac{\sqrt{\mu[m_l](1-\mu[m_l])}}{\sqrt{T}}$.

  It is clear that the structure of the linear response in the case of spatio-temporal constraints is quite a bit more complex than the case $D = 0$ (see Eq. (2.21)). Actually, for $D = 0$, all correlations $C_{m_l\, m_k}(n)$ vanish for $n > 0$ (distinct times are independent).

- $\mathcal{P}(\mathcal{H})$ is a convex function of $\boldsymbol{\lambda}$. As a consequence, if there is a set of $\boldsymbol{\lambda}$ value, $\boldsymbol{\lambda}^*$, such that .

$$\frac{\partial \mathcal{P}[\mathcal{H}]}{\partial \lambda_l^*} = \mu[m_l] = C_l, \tag{2.40}$$

then this set is unique. Thus, the solution of the variational problem (Eq. 2.37) is unique.

Basically, Eq. 2.38, 2.39 and 2.40, tell us that techniques based on free energy expansion in spatial models can be extended as well to spatio-temporal cases, where the free energy is replaced by the topological pressure. Obviously, estimating (not to speak of computing) the topological pressure can be a formidable task. Although, the transfer matrix technique allows the computation of the topological pressure, the use of this method for large N is hopeless (see section 2.5). However, techniques based on periodic orbit expansion could be useful [Cofré and Cessac, 2013]. Additionally, cumulative expansions of the pressure, Eq. 2.38 and 2.39 corresponding to the two first orders, suggest that extension of methods based on free energy expansion could be used. In addition to the works quoted above, we also think of constraint satisfaction problems by Mézard and Mora [Mézard and Mora, 2009] and approaches based on Bethe free energy [Welling and Teh, 2003]. Finally, as we checked, the properties of spatio-temporal Gibbs distributions allows to extend the parameters estimation methods developed for the spatial case in [Dudík et al., 2004, Broderick et al., 2007] to spatio-temporal distributions (we explain this extension in detail in the section 3.6).

2.4 Maximum Entropy modeling process in practice

In this section, we show what are the models we usually use, the process from obtaining the spike train until models evaluation. For that, we give first an idea about the models/potentials shapes and then we apply on some examples.

2.4.1 Potential's typical canonical forms

Canonical potentials are those which are typically used to shape typical Maximum Entropy models. Bernoulli and Ising are the simplest ad-hoc models and they have been respectively defined in Eq. 2.22 and 2.23. The triplets model is also an important canonical model and has been studied by [Ganmor et al., 2011a]. Its potential reads:

$$\mathcal{H} = \sum_i \lambda_i \omega_i(0) + \sum_{i,j} \lambda_{ij} \omega_i(0)\omega_j(0) + \sum_{i,j,k} \lambda_{ijk} \omega_i(0)\omega_j(0)\omega_k(0) \qquad (2.41)$$

where ω_{ijk} corresponds to the triplet interactions, i.e., the 3 neurons i, j and k fires in the same time.

Finally, the potential of pairwise models with delay, where temporal interactions take place, reads:

$$\mathcal{H} = \sum_{l} \lambda_i \omega_i + \sum_{i,j,\delta} \lambda_{ij}^{\delta} \omega_i(0)\omega_j(\delta), \tag{2.42}$$

where δ is the variable that rises the temporal effect. The pairwise monomial $(\omega_i(0)\omega_j(1))$ corresponds to the fact that the neuron i fires 1 time-bin after the neuron j, and so on. The maximum δ that we can have is related to the Range of the potential, i.e., $\delta \in \{0, R-1 = D\}$.

Computing the Kullback-Leibler divergence in practice

In the spatio-temporal case, computing the KLD using Eq. 2.8 is computationally impossible. For that, we use another tractable version:

$$d_{KL}\left(\nu, \mu_{\boldsymbol{\lambda}}\right) = \mathcal{P}\left[\boldsymbol{\lambda}\right] - \nu\left[\mathcal{H}\right] - \mathcal{S}\left[\nu\right]. \tag{2.43}$$

Replacing ν by $\pi_{\omega}^{(T)}$, we have:

$$d_{KL}\left(\pi_{\omega}^{(T)}, \mu_{\boldsymbol{\lambda}}\right) = \mathcal{P}\left[\boldsymbol{\lambda}\right] - \pi_{\omega}^{(T)}\left[\mathcal{H}\right] - \mathcal{S}\left[\pi_{\omega}^{(T)}\right]. \tag{2.44}$$

For small networks, we can compute the exact value of the divergence because we know the exact value of the pressure and the monomials averages. The entropy $\mathcal{S}[\pi_{\omega}^{(T)}]$ is computed directly from the raster using the technique of Strong ([Strong et al., 1998]).

Computing the exact divergence for $NR > 20$ is impossible since we cannot compute the exact values of the pressure and the monomials averages. For that, section 3.5.1 is dedicated to present a method to compute the KLD for larger networks.

2.4.2 From the spike train to the model

The process of inferring the probabilistic model from a spike train consists on the following steps (Figure 2.6):

- Data preparation: one chooses the neurons and the binning.
- Model choice: one chooses one of the ad-hoc models presented above. Note that thanks to *EnaS* which we will present in chapter 4, one can chooses one of those models as well as define customized models.

- Computing the empirical distribution (this will set up the constraints).
- Fitting the parameters $\boldsymbol{\lambda}$: here we go through a process of minimization of the KLD criterion. For small network we have used the method presented in [Vasquez et al., 2010]. We shall see in the next chapter another method which is more suitable for larger networks.
- Evaluation via confidence plots and KLD.

2.5 The advantages and limits of transfer matrix method

The advantage of the transfer matrix technique method is that it is mathematically exact: given a potential $\mathcal{H}$, it gives the Gibbs distribution and topological pressure without computing a partition function; given the parametric form (Eq. 2.4) where the parameters λ_ks has to be determined ("learned"), it provides the unique solution. On numerical grounds, this method provides an optimal estimation, in the limits of the error made when observing the observables empirically, this error being characterized by the central limit theorem. Its main drawback is that the transfer matrix $\mathcal{L}$ has 2^{NR} entries ! Although, most of those entries are zero (2^N non zero entries per row, thanks to the compatibility conditions) it is simply too huge to handle cases where $NR > 24$.

Focusing thus on the huge number of states in the set of blocks, it is clear that any method requiring the extensive description of the phase space fails as NR grows. Additionally, while the accessible phase space is huge, the *observed* phase space (e.g. in an experimental raster) is rather small. Several strategies exist to avoid the extensive description of the phase space. Here, we propose an approach based on Montecarlo sampling.

The idea is the following. Given a potential $\mathcal{H}$ we find a strategy to approximately compute the average $\mu[\, m_l \,]$ of observables m_l under the Gibbs distribution μ, using a statistical Montecarlo sampling of the phase space. For that purpose, the algorithm generates a raster following the statistics defined by the potential $\mathcal{H}$, and computes the observables on this artificial raster. Thanks to the estimation of the observables, the parameters of the model (λ_ks) can be found by modifying their values to minimize iteratively the distance between the values of the observables estimated on the real raster, and the values estimated with the Montecarlo sampling. Powerful algorithms exist for that, taking into account the uncertainty on the empirical averages ruled by the central limit theorem ([Dudík et al., 2004] and

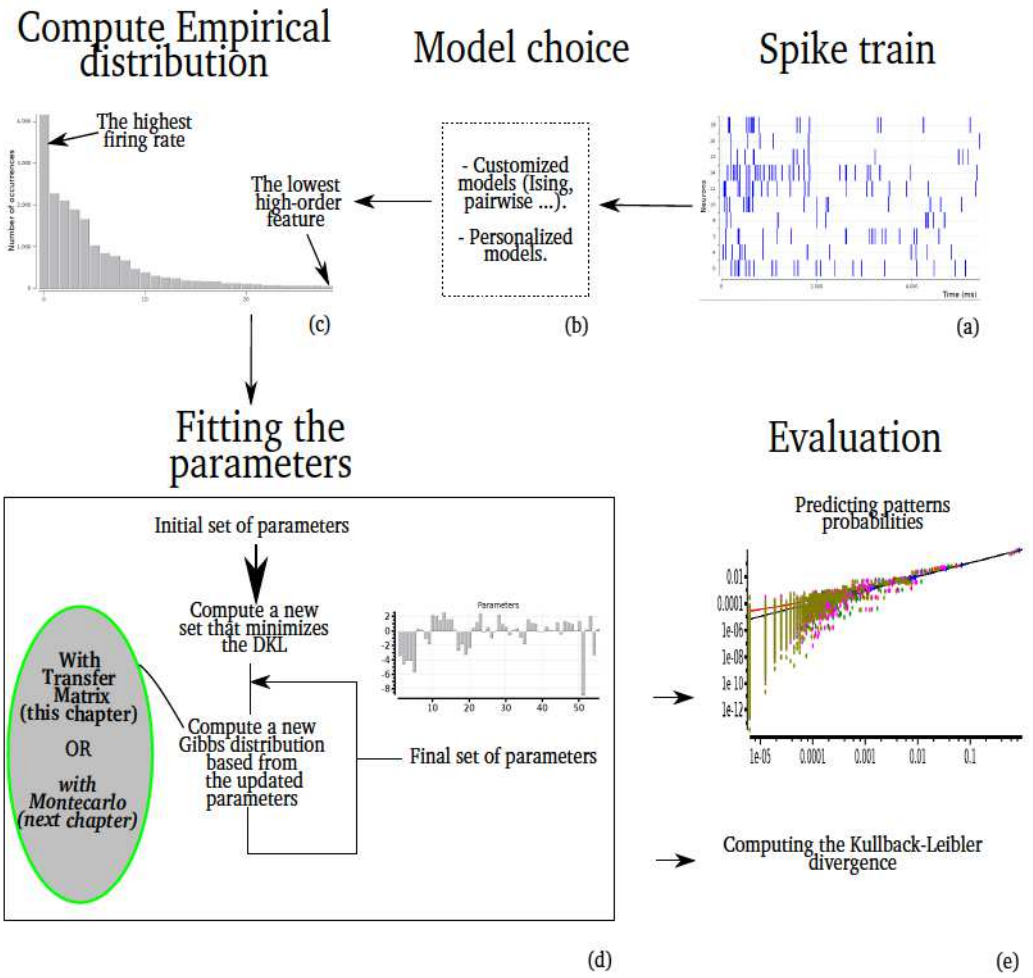


Figure 2.6: Given a spike train (a) and a model (which represents the constraints/features/monomials) (b), the first step is to compute the empirical averages of the monomials (c). The next step is to compute the parameters. (c) To find the parameters, we iterate from an initial condition in order to update the parameters step by step. Once we update the parameters based on some criterion, we compute the target probability distribution that corresponds to the found parameters. Computing this probability distribution could be performed whether via the transfer matrix (explained in chapter 2) or with Montecarlo that will be explained in this chapter. The last step is the evaluation (d). We compute the probabilities of all the possible pattern (until an limited depth (1-5 time steps typically) in order to compare the predictions of the models with the empirical measures.

[Broderick et al., 2007])). These algorithms initially developed for the spatial case have been extended by us as explained in the Chapter 3.

2.6 Conclusion

The methods presented in the literature are confronted with three main limitations:

1. Limitation to purely spatial models in the case of large networks [Tkačik et al., 2009].
2. Limitation to small network in the case of Maximum Entropy models with spatio-temporal constraints.

Hence, non of the existing frameworks is able to compute Maximum Entropy models for large networks with spatio-temporal constraints. We will define large scale networks as those where $N \times R$ is bigger than 20. As a consequence, we wanted to develop a new technique that is able in the same time to:

- Analyze large scale networks (Solution for 1).
- Take the spatio-temporal constraints and the instantaneous correlations between neurons into account (Solution for 2).

In the next chapter, we will show a method we developed, based on Montecarlo principle in order to be able to analyze large network taking into account instantaneous correlations as well as memory effects.

Chapter 3

Modeling large scale spiking data sets with spatio-temporal constraints using Montecarlo method

Contents

In this chapter, we present an approach based on Montecarlo sampling to analyze large scale spike trains. We consider the case when $NR > 20$. In this approach, the generation of the target distribution is based on a Montecarlo algorithm (section 3.2) with a parallelized version (section 3.3) on multi-processors computers. In section 3.4 we show validation test for the Montecarlo algorithm as well as the computation time in section 3.5. The parameters fitting is based on minimizing an equivalent criterion of the Kullback-Leibler divergence (section 3.6). Finally, we present the results on synthetic data in section 3.7 and finish with concluding remarks on the method.

3.1 Results on synthetic data

3.2 Computing spatio-temporal Gibbs distributions with Montecarlo method

3.2.1 The Montecarlo-Hastings algorithm

The Montecarlo-Hastings method consists of sampling a target probability distribution μ by constructing a Markov chain whose invariant probability is μ [Hastings, 1970]. The transition probability of this Markov chain, between two states $\omega^{(1)}$ and $\omega^{(2)}$ is:

$$P[\omega^{(1)}|\omega^{(2)}] = \max\left(\frac{Q(\omega^{(1)}|\omega^{(2)})}{Q(\omega^{(2)}|\omega^{(1)})}\frac{\mu\left[\,\omega^{(2)}\,\right]}{\mu\left[\,\omega^{(1)}\,\right]}, 1\right). \tag{3.1}$$

The function $Q()$ can have different forms, allowing in particular to boost the convergence rate of the algorithm. Such specific forms are highly dependent on the form of $\mathcal{H}$, and there is no general recipe to determine Q, given

$\mathcal{H}$. The contribution of Q cancels in Eq. 3.1 whenever Q is symmetric ($Q(\omega|\omega') = Q(\omega'|\omega)$). We make this assumption in the sequel. Practically, we take Q as the uniform distribution corresponding to flipping one spike at each iteration of the method. This is not necessarily the best approach as e.g. clusters update would be more efficient. But we do not know about any cluster algorithm for spatio-temporal potentials.

In classical Montecarlo approaches in statistical physics, the normalization factor of the Gibbs distribution, the partition function, cancels when computing the ratio of two blocks probabilities $\frac{\mu\left[\omega^{(2)}\right]}{\mu\left[\omega^{(1)}\right]}$. The situation is different in the presence of spatio temporal constraints, as shown in Eq. 2.32: "boundary terms" $L\left(\omega_0^{D-1}\right)$, $R\left(\omega_{n-D+1}^{n}\right)$ remain. Actually, the same would hold in statistical physics problem with spatial interactions if one were to compare the probability of bulk spin-chains with distinct boundary conditions.

This problem can however be circumvented thanks to the following remarks:

1. If one compares the probability of two blocks $\omega^{(1)}, \omega^{(2)}$ of range $n > 2D+1$, with $\omega_0^{D-1,(1)} = \omega_0^{D-1,(2)}$ and $\omega_{n-D+1}^{n,(1)} = \omega_{n-D+1}^{n,(2)}$ then Eq. 2.32 reads:

$$\frac{\mu\left[\omega_0^{n,(2)}\right]}{\mu\left[\omega_0^{n,(1)}\right]} = \frac{\frac{e^{\mathcal{H}\left(\omega_0^{n,(1)}\right)}}{s_{\boldsymbol{\beta}}^{n-D+1}} R\left(\omega_{n-D+1}^{n,(1)}\right) L\left(\omega_0^{D-1,(1)}\right)}{\frac{e^{\mathcal{H}\left(\omega_0^{n,(2)}\right)}}{s_{\boldsymbol{\beta}}^{n-D+1}} R\left(\omega_{n-D+1}^{n,(2)}\right) L\left(\omega_0^{D-1,(2)}\right)} \tag{3.2}$$

   $s_{\boldsymbol{\beta}}^{n-D+1}$ is the same in the numerator and the denumerator. However, $R\left(\omega_{n-D+1}^{n,(2)}\right)$ and $R\left(\omega_{n-D+1}^{n,(1)}\right)$ are not the same. Nor $L\left(\omega_{n-D+1}^{n,(2)}\right)$ and $L\left(\omega_{n-D+1}^{n,(1)}\right)$ are the same. If we keep those 4 terms in the ratio, we will be obliged to compute them during the Montecarlo process. The tricks we used to circumvent this problem is to keep blocks ω_0^{D-1} and ω_{n-D+1}^{n} unchanged when updating the states of the Montecarlo spike train. As a consequence, we will have:

$$R\left(\omega_{n-D+1}^{n,(1)}\right) = R\left(\omega_{n-D+1}^{n,(2)}\right)$$

$$L\left(\omega_{n-D+1}^{n,(1)}\right) = L\left(\omega_{n-D+1}^{n,(2)}\right)$$

As a consequence, we will have:

$$\begin{aligned} \frac{\mu\left[\omega_0^{n,(2)}\right]}{\mu\left[\omega_0^{n,(1)}\right]} &= \frac{e^{\mathcal{H}\left(\omega_0^{n,(1)}\right)}}{e^{\mathcal{H}\left(\omega_0^{n,(2)}\right)}} \\ &= e^{\Delta\mathcal{H}(\omega^{(1)},\omega^{(2)},0,n)} \end{aligned} \tag{3.3}$$

with

$$\Delta\mathcal{H}(\omega^{(1)},\omega^{(2)},0,n) = \mathcal{H}\left(\omega_0^{n,(1)}\right) - \mathcal{H}\left(\omega_0^{n,(2)}\right).$$

Thus, the Montecarlo transition probability (Eq. 3.1) is only expressed as a difference of potential of the two blocks.

2. $\Delta\mathcal{H}(\omega^{(1)},\omega^{(2)},0,n) = \sum_{k=1}^{K}\lambda_k\Delta m_l(\omega^{(1)},\omega^{(2)},0,n)$, with:

$$\Delta m_l(\omega^{(1)},\omega^{(2)},0,n) = \sum_{l=0}^{n-D}\left[m_l\left(\omega_l^{D+l,(2)}\right) - m_l\left(\omega_l^{D+l,(1)}\right)\right].$$

Since the m_ls are monomials, many terms $m_l(\omega_l'^{l+D}) - m_l(\omega_l^{l+D})$ cancel. Assuming that we flip a spike at position (k,t), $k \in \{1,\ldots,N\}$, $t \in \{D, n-D\}$, we have indeed:

$$\Delta m_l(\omega^{(1)},\omega^{(2)},0,n) = \sum_{l=t-D}^{t}\left[m_l\left(\omega_l^{D+l,(2)}\right) - m_l\left(\omega_l^{D+l,(1)}\right)\right]$$

Since the difference $m_l\left(\omega_l^{D+l,(2)}\right) - m_l\left(\omega_l^{D+l,(1)}\right) \in \{-1,0,1\}$, the computational cost of $\Delta m_l(\omega^{(1)},\omega^{(2)},0,n)$ is minimal if one makes a list of monomials affected by the flip of spike (k,r), $r = 0,\ldots D$.

Figure 3.1 details the computing of the theoretical probability distribution with the Montecarlo algorithm. We begin with a random spike train of size N neurons and length N_{times} time-steps. We choose one neuron randomly in the interval $[R, T-R]$ and we flip its state (if the neuron fired we make it silent and if it was silent we make it firing). This process is to judge which state (firing or silent) is more likely to make the theoretical probability distribution of the raster closer to the empirical probability. We check this by computing the $\frac{\mu\left[\omega_0^{n,(2)}\right]}{\mu\left[\omega_0^{n,(1)}\right]}$. If this value is bigger that a random number in $[0,1]$, we accept the change, otherwise, we reject it. We repeat this step N_{flip} times.

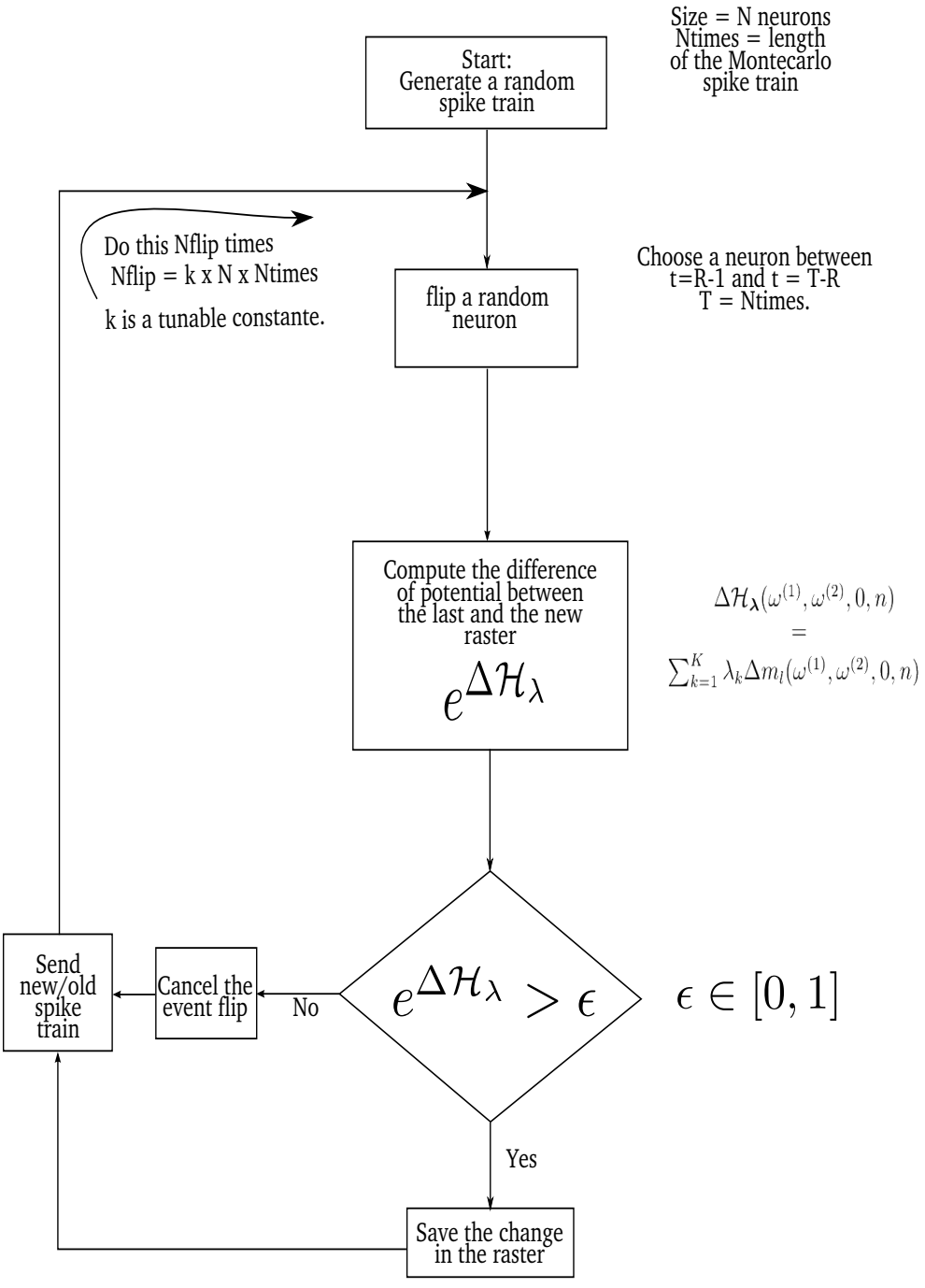


Figure 3.1: The generation of a spike train whose distribution is close to the empirical distribution.

3.2.2 Convergence rate

The goal of Montecarlo-Hastings algorithm is to generate a sample of a target probability obtained by iteration of the Markov chain defined by Eq. 3.1. In our case, this sample is a raster ω_0^{T-1}, distributed according to a Gibbs distribution μ. Call N_{flip} the number of iterations ("flips" in our case) of the Montecarlo algorithm. As $N_{flip} \to +\infty$ the probability that the algorithm generates a raster ω_0^{T-1} tends to $\mu\left[\omega_0^{T-1}\right]$. Equivalently, if one generates N_{seed} rasters and denote $\#\left(\omega_0^{T-1}\right)$ the number of occurrences of a specific bloc ω_0^{T-1}, then:

$$\lim_{N_{seed}\to+\infty}\lim_{N_{flip}\to+\infty}\frac{\#\left(\omega_0^{T-1}\right)}{N_{seed}} = \mu\left[\omega_0^{T-1}\right].$$

The convergence is typically exponential with a rate depending on $\mathcal{H}$.

Now, the goal here is to use a Montecarlo raster to estimate $\mu\left[m_l\right]$ by performing the empirical average $\pi_\omega^{(T)}\left[m_l\right]\omega$ on that raster. However, as explained in section 2.1.4, even if the raster is distributed according to μ (corresponding thus to taking the limit $N_{flip} \to +\infty$) the empirical average $\pi_\omega^{(T)}\left[m_l\right]\omega$ is not equal to $\mu\left[m_l\right]$, it converges to $\mu\left[m_l\right]$ as $T \to +\infty$, with an exponential rate. More precisely, the probability that the difference $\left|\pi_\omega^{(T)}\left[m_l\right]\omega - \mu\left[m_l\right]\right|$ exceeds some $\epsilon > 0$ behaves like $\exp(-T \times I(\epsilon))$ where $I(\epsilon)$, called large-deviations rate, is the Legendre transform of the topological pressure [Chazottes and Keller, 2008].

When T is large we have:

$$\mu\left[\left|\pi_\omega^{(T)}\left[m_l\right]\omega - \mu\left[m_l\right]\right| > \epsilon\right] \simeq \exp\left(\frac{-T \times \epsilon^2}{\sigma(m_l)}\right) \tag{3.4}$$

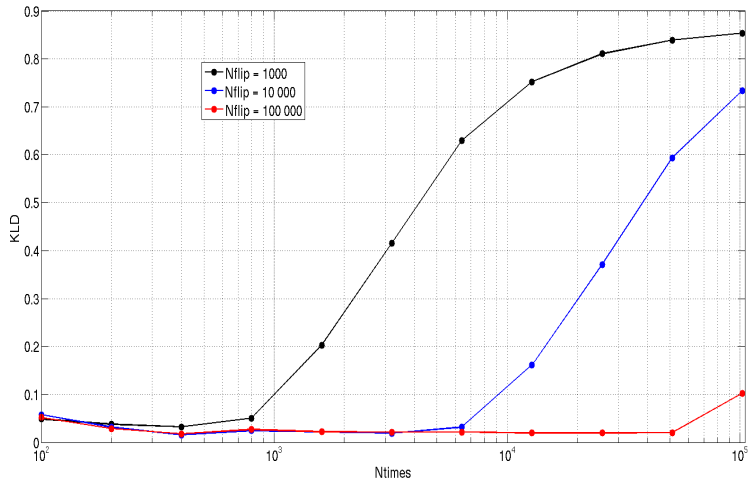
where $\sigma(m_l) = \sqrt{\mu\left[m_l\right]\left(1-\mu\left[m_l\right]\right)}$ is the mean-square deviations of m_l.

As a consequence, to obtain the exact average $\mu\left[m_l\right]$ from our Montecarlo algorithm we would need to take the limits:

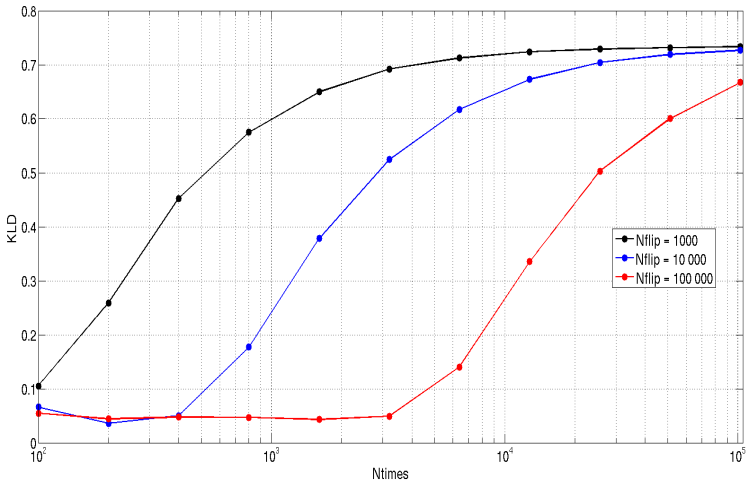
$$\lim_{T\to+\infty}\lim_{N_{seed}\to+\infty}\lim_{N_{flip}\to+\infty}\frac{\#\left(\omega_0^{T-1}\right)}{N_{seed}}, \tag{3.5}$$

in *that order*: they do not commute. A prominent illustration of this point is illustrated in Figure 3.2.

For notation homogeneity we note from now on $T-1 \equiv N_{times}$ for the raster length. When dealing with numerical simulations with a finite number of sample, the goal is to minimize the probability that the error is bigger than a real number ϵ, by suitable choice of:



(a) 3 neurons



(b) 5 neurons

Figure 3.2: Error as a function of Ntimes, for several values of Nflip (1000, 10000, 100000). (a) N = 3; (b) N = 7. For the same Ntimes value, the error committed on the target distribution decreases when Nflip increases. Likewise, when we look at one curve that describes how the error increases as Ntimes increases for a fixed Nflip value, this is but a justification that the value of Nflip is not enough for such sample size. From [Nasser et al., 2013].

- The raster length: $T-1=N_{times}$.
- The number of flips: N_{flip}.
- The number of seed: N_{seed}.

Let us now establish a few relations between those parameters. First, it is somewhat evident that N_{flip} must be at least proportional to $N \times N_{times}$ in order to give a chance to all spikes in the raster to be flipped at least once. This criterion respects the order of limits in Eq. 3.5.

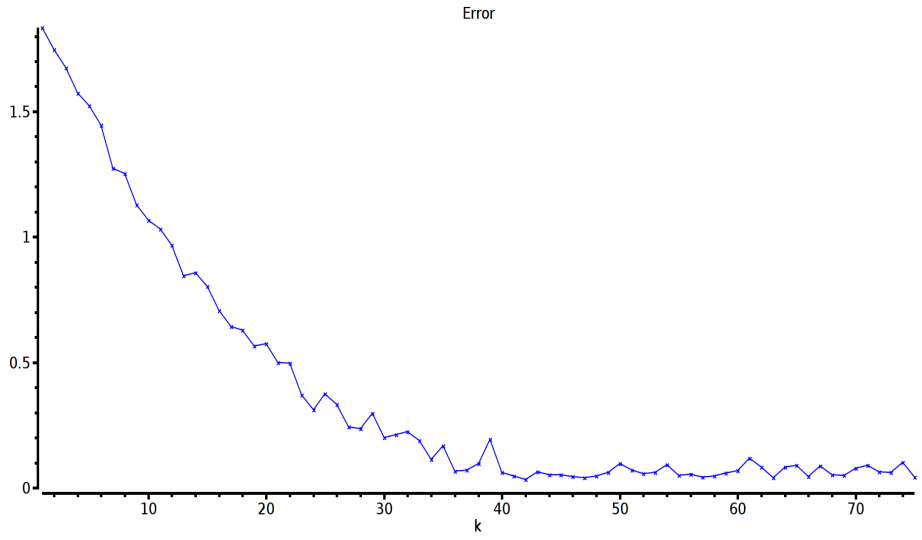
Since μ is ergodic one can in principle estimate the average of observables by taking $N_{seed}=1$ and taking N_{times} large. However, the larger N_{times} the larger N_{flip} and too big N_{times} leads to too long simulations. On the opposite, one could generate a large number N_{seed} of raster with a small N_{times}. This would have the advantage of reducing N_{flip} as well. However, the error (Eq. 3.4) would then be too large. So, one needs to find a compromise: N_{times} large enough to have small Gaussian fluctuations (Eq. 3.4) and small enough to limit N_{flip}. Then, by increasing N_{seed}, one approaches the optimal bound on fluctuations given by Eq. 3.4. In our simulations, the number of flips is determined using the following formula:

$$N_{flip} = k \times N \times N_{times} \tag{3.6}$$

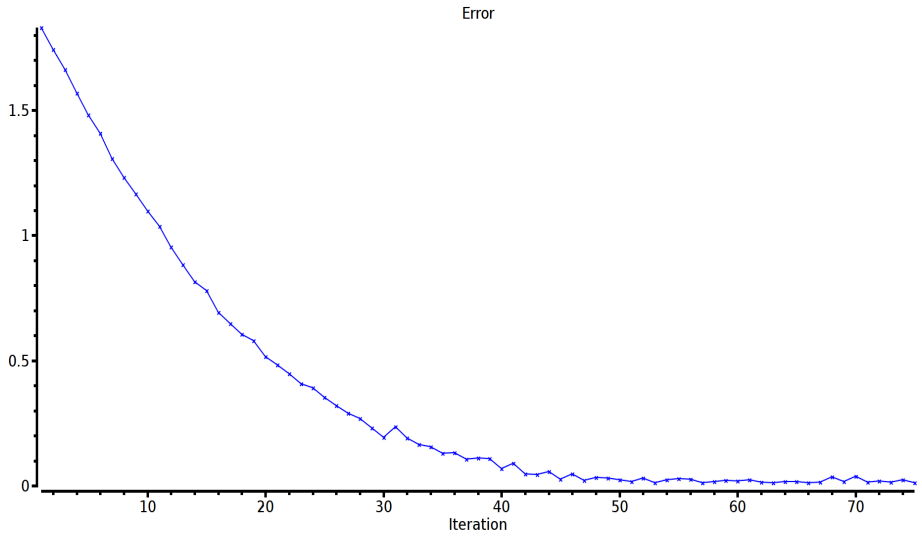
where k is a constant to be determined by the user. We recommend k to be bigger than 30. In order to show the effect of k, we did the following experience: we fitted a pairwise model of a range $R=2$ on a real spike train of 5 neurons with the transfer matrix method which gives us an exact solution for the parameters and the expected monomials averages. From this set of parameters, we computed the target distribution using the Montecarlo algorithm for several values of k (we tested with $N_{times}=10^5$ and 10^6). Afterwards, we computed the error[1] (as a function of k) between the reference (exact) values of monomials averages (those of the transfer matrix) and the monomials averages computed with the Montecarlo algorithm. We observe that this error decreases as k increase (Figure 3.3).

In fact, if $k=1$, the events in the Montecarlo spike train will be toggled uniformly since we are performing randomly $N \times N_{times}$ toggles in a matrix of size $N \times N_{times}$. Giving the chance to all events to be toggled only once time does not allow the distribution to converge. Each of the events should be toggled sufficiently many times because their states depend on the states of the surrounding events. For instance, if an event $\omega_i(a)$ was toggled and

[1] We used the Hellinger Distance (defined in 3.6.3 as an error criterion.



(a) $N_{times} = 100\ 000$



(b) $N_{times} = 1000\ 000$

Figure 3.3: The error between a target distribution computed with the transfer matrix and Montecarlo as a function of k. On a set of 5 neurons, we computed the parameters using the transfer matrix process. From the exact set of parameters, we computed the target distribution with Montecarlo for $k = 1 \ldots 75$ in two cases: (a) $N_{times} = 100\ 000$ and (b) $N_{times} = 1000\ 000$. The error decreases as k increase. After $k = 40$ we observe a steady state, e.g., the increase of k does not decrease the error. By consequence, it is useless to compute the distribution with a $k > 40$.

its new configuration is accepted based on other two events $\omega_j(b)$ and $\omega_k(c)$; when one of those events changes afterwards if they have been chosen to be toggled, the configuration of $\omega_i(a)$ will not be necessarily correct. Hence, we would toggle it another time in order to see which configuration to keep. Ideally, if $k \longrightarrow \infty$, the Montecarlo algorithm would give the same results as the transfer matrix. However, Figure 3.3 shows that $k = 30$ is good enough.

3.3 Parallelization of Montecarlo algorithm

Computing the target distribution is computationally heavy and time consuming. We decided to circumvent this obstacle by parallelizing the computation during the Montecarlo process. The parallelization is possible thanks to the OpenMP framework that allows many processors on the same machine to work in parallel and in the same time on one data set. OpenMP or Open Multi-Processing is an API (application programming interface) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran, on most processor architectures and operating systems. Another parallelization framework exists, the MPI (Message Passing Interface) that is possible only on clusters. A computer cluster consists of a set of connected computers that work together so that in many respects they can be viewed as a single system. We found out that the MPI is less advantageous than the OpenMP, for the reasons that will be explained later.

3.3.1 Parallelization over the spike train size with OpenMP

The idea is based on decomposing the Montecarlo spike train to N_p parts, where N_p is the number of available processors on the computer. Given that the Montecarlo spike train is of length N_{times}, each processor will be responsible on updating the raster over $\frac{N_{times}}{N_p}$ time steps. Hence, we will have N_p target distributions generated from a set of parameters $\boldsymbol{\lambda}_l$. By dividing the spike train into N_p parts, the memory although shared, but each processors has the right to access only one part of the spike train (Figure 3.4). Each processor applies the Montecarlo update independently from the other on its dedicated piece of raster.

The critical flips During flipping, and since choosing the events is random, it may happen that two neighbor processors toggle events at their common boundaries. Updating one of the events depends on its surrounding, which should be not changing. In the case where the surrounding is changing, the result of the update is wrong because updates from the processors are not

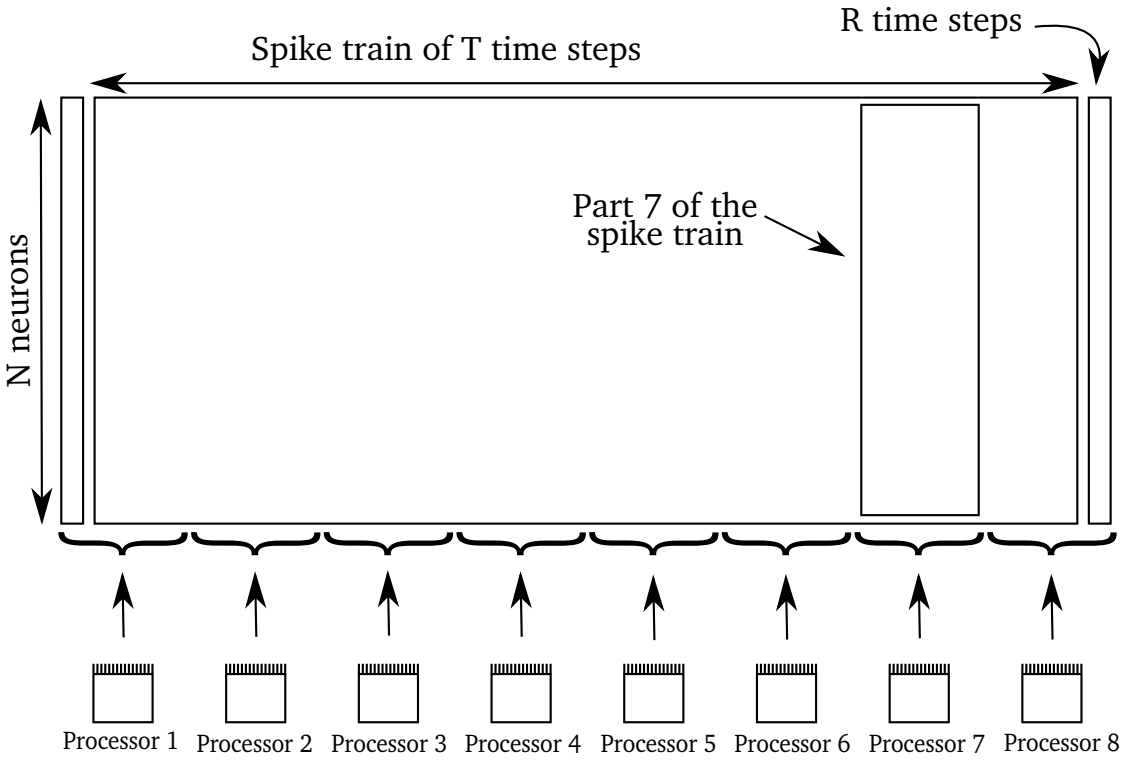


Figure 3.4: An example of parallel processing over OpenMP with a computer of 8 processors. Each of the processors accesses exclusively one part of the spike train, that is stocked in a shared memory space. Note that we have to add R time step (random events) before the beginning and after the end of the spike train because of the critical boundaries. The critical boundaries presented here are those at the beginning and the end of the Montecarlo spike train. Another example of critical boundaries that exist between processors is presented in Figure 3.5.

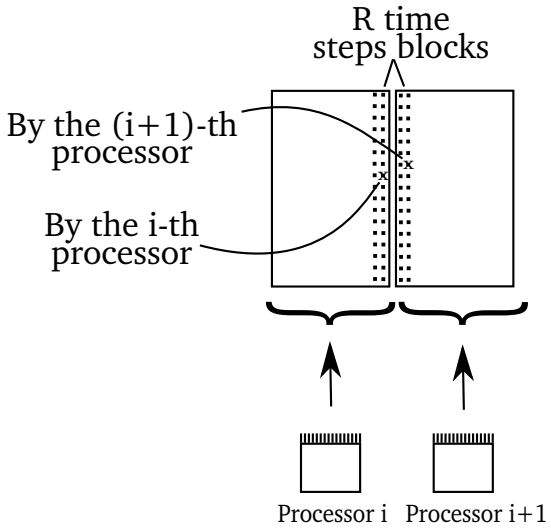


Figure 3.5: Toggling event at the critical boundaries of two adjacent parts of the spike train gives wrong results. However, due its low probability of happening and the big size of the Montecarlo spike train, we consider that it does not affect the result of the target distribution.

independent. This is because, each time we toggle a neuron, we compute the change in energy (Eq. 3.1), which depends from the surrounding events. If the surrounding events are changing in the same time, computing the Eq. 3.1 will be based on wrong events. The probability that this case happens is low $\frac{1}{(NT)^2}$. It can nevertheless affects the results if T is too small.

3.3.2 Parallelization over the number of flips with OpenMP

Parallelizing over N_{flip} means that all the processors will have the access to the whole Montecarlo spike train (in opposite to parallelizing over N_{times} where each processor has access to one part of the spike train) but they share the parallelized task which is toggling events. So, each processor would be responsible of toggling $\frac{k \times N \times Ntimes}{N_p}$ times the events. This case would be more advantageous if we have more processors.

This implies that the N_p processors will flip simultaneously the events and in order to accept/or not the toggling. Unfortunately, this algorithm is not true. This case of parallelization allows any of the N_p processors to (simultaneously) toggle randomly chosen events in the spike train. The probability of choosing any event is $p = \frac{1}{NT}$. This probability is small. The probabil-

ity that two processors choose the same (or neighbor) event(s) is then p^2. When the number of processors increases, the probability of accessing the same (or neighbor) event(s) in the same time increases. For instance, if there are 3 processors, the probability is $p^3 + C_3^2 p^2$ ...n processors, the probability is $p^{N_p} + C_{N_p}^{N_p-1} p^{N_p} + \ldots$. As we explained before that toggling instantaneously two neighbor events gives wrong result, then the bigger the number of processors, the bigger the probability to have wrong computations during Montecarlo. The extreme case, where the number of processors is equal to the size of the spike train, the probability of having wrong computations is equal to 1.

3.3.3 Parallelization with MPI

We performed also the parallelization on our cluster using MPI, where we can have much more bigger number of processors than on personal computers (2, 4 or 8 processors). Our cluster is called Nef Torque (Figure 5.15) `https://nef-services.inria.fr/` at INRIA Sophia-Antipolis research center. It contains 37 nodes that consists on a total of 876 processors. We found out that the time needed to finish the computation decreases with the number of processors. However, after some critical number of processors, the computation time begins to increase and the use of parallel computing on the cluster is not advantageous anymore. The reason is that, the more the number of processors used for the computing, the more the number of exchanged messages between processors. As a consequence, the processors spend more time to send messages to each other instead of performing the computing.

3.3.4 OpenMP on clusters

We opted for an alternative solution by taking benefits from the cluster computing power and the OpenMP algorithm where the memory is shared and processors do not need time to communicate. As we precised lately, running the algorithm on personal computers with OpenMP is restricted with the limited number of processors. We found that the alternative could be running the Montecarlo process on the cluster using OpenMP.

3.4 Validation of the Montecarlo algorithm

We generate a potential with a known probability distribution (the monomials are generated randomly, but they are known). For each potential, the monomial set corresponds to rates and higher order monomials. The higher

Figure 3.6: The Nef Torque cluster, at INRIA Sophia-Antipolis research center. A cluster with 37 nodes and 876 processors.

order monomials are created from randomly generated masks. The number of higher order monomials is set according to the range of of the potential (we used usually $5 \times R$ higher order and spatio-temporal monomials).

In order to test the performance of Montecarlo, we do the following test: given the known potential, we compute the target distribution that corresponds to this potential with the transfer matrix method on one hand and with the Montecarlo method on the other hand. In fact, with the transfer matrix method we get the exact target distribution used as a reference with which we compare the results of Montecarlo. This comparison limits us to a small number of neurons. Figure 3.7 shows the comparison of the target distribution features averages.

Finally, we tested the convergence of the algorithm by computing the Kullback-Leibler divergence in term of the sample size (the length of the spike train generated with Montecarlo). Figure 3.8 show that the Kullback-Leibler divergence vanishes as the length of the sample increases. This follows the CLT theorem and shows the generating the target distribution with Montecarlo convergences. We performed the test in the case of both dense and sparse spike train. Dense spike train show higher firing rate than sparse ones, both are explained in details in the section 3.7.1.

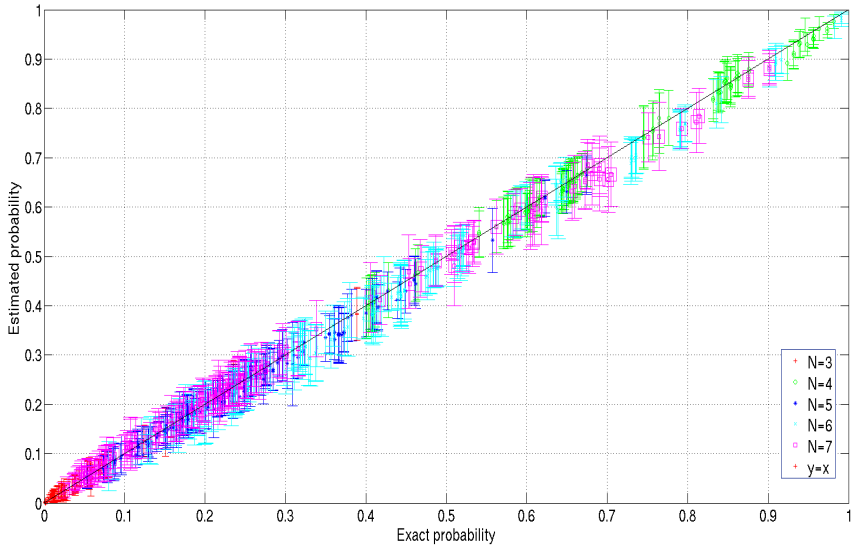
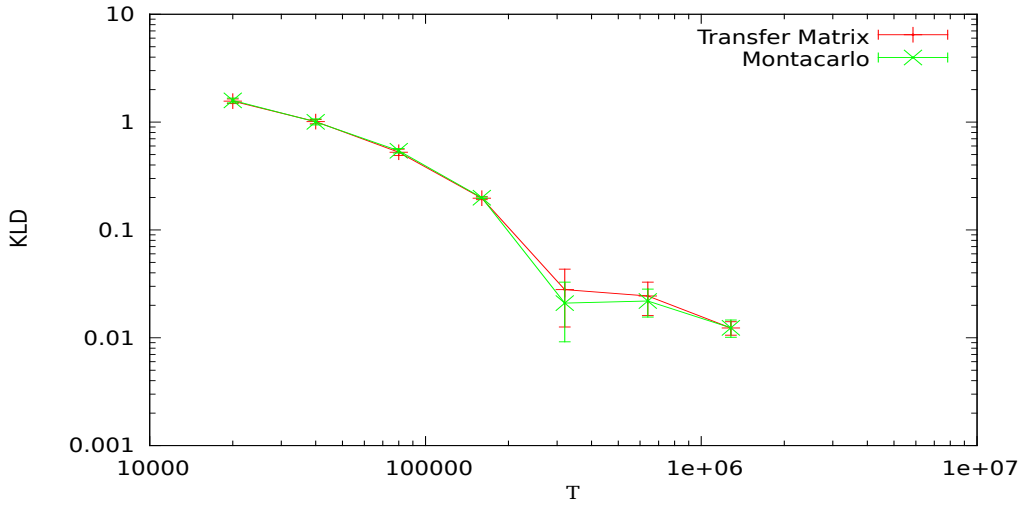


Figure 3.7: We generated random potential with several number of neurons. We computed the monomials averages with the the transfer matrix method (on the abscissa) and the Montecarlo method (on the ordinate). Note that in the Montecarlo case, we computed 10 target distribution is each case and the ordinate axis show the point with mean and standard deviation over the 10 target distribution. The $y = x$ axis show that the Montecarlo algorithm for computing the Gibbs distribution is close to the transfer matrix technique.

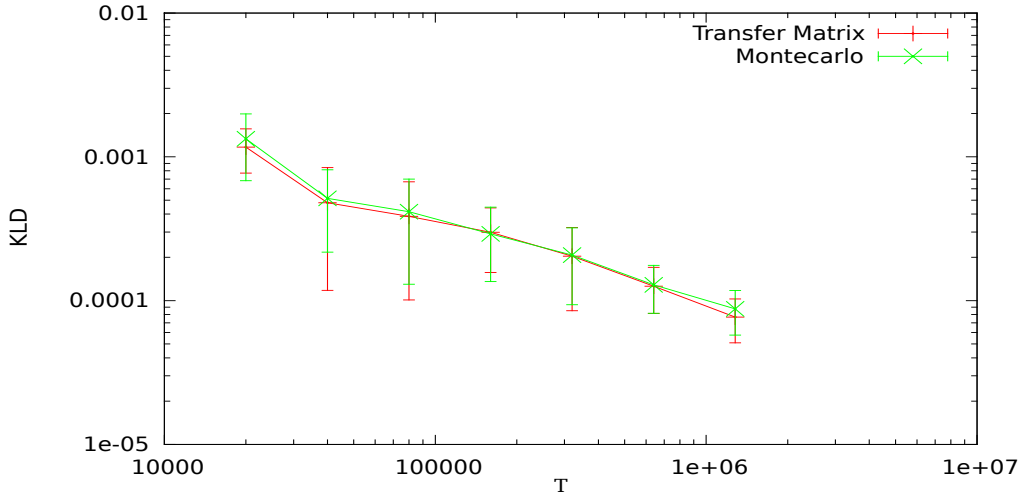
3.5 Computation time

Let us call, respectively, t_t and t_{mc}, the time needed to compute a target distribution with the transfer matrix and Montecarlo. In order to determine how t_t and t_{mc} evolves with respect to N, we did the following experiment: given a number of neurons N (from $N = 3$ until $N = 8$), we generate L random monomial of range 3 and we associate random parameters for the distribution. L depends on the number of neurons. For each set, we generate N rates monomials, $\frac{N(N-1)}{2}$ pairwise and $\frac{N(N-1)(R-1)}{2}$ monomials with delay. From the set of parameters, we compute for each N a target distribution with the transfer matrix on one hand and with the Montecarlo on the other hand. We plot the time taken for each neuron's set in a log scale, Figure 3.9.

If we compare the two computation times in a log scale (Figure 3.9), we observe that t_{mc} increases linearly and t_t increases exponentially as N gets larger. This means, also, that t_t increases faster that, i.e., the difference



(a) Dense Gibbs distributions.



(b) Sparse Gibbs distributions.

Figure 3.8: The decay of the KLD with the length of the sample size. (a) Dense. (b) Sparse (explained in detail in section 3.7.1). The red and green curves show the decay of the DKL using respectively the transfer matrix and Montecarlo methods. Note that, for each point, we generate 10 randoms distribution and we compute the mean and variance of the error over the 10 samples. The similar decay of DKL both with Montecarlo and the Transfer matrix shows the efficiency of the Montecarlo sampling for generating the target distribution.

between the two computation times gets very large for larger N. As shown in Figure 3.9, apart from 7 neurons, t_t begins to increase dramatically. t_{mc} is also still growing, but with a lower speed.

We would like to give an idea about the computational complexity for computing a target distribution with Montecarlo and determine the relation between t_{mc} and the computation variables such as $N, N_{times}, N_{flip} \ldots$. As Figure 3.1 shows, we have a loop over which the processor runs. The iterations of this loop are equal to $k \times N \times N_{times}$. At each iteration, we compute the value of $e^{\Delta\mathcal{H}}$, call it $t_{\Delta\mathcal{H}}$. Then, the total time needed is:

$$t_{mc} = k \times N \times N_{times} \times t_{\Delta\mathcal{H}} \tag{3.7}$$

where $t_{\Delta\mathcal{H}}$ is a function of the number of monomials L. There is no explicit form for $t_{\Delta\mathcal{H}} = f(L)$. However, we know that $f(L)$ is not linear in L nor in N because of the reasons explained below. Since k and N_{times} are usually constante, we will express then the computational complexity to compute the target distribution from now on as: $\mathcal{O}(f(N, L))$. **The 2 reasons for which $f(L)$ is not linear in N nor in L:**

- L **is not proportional to** N: the dimensions of the canonical model shapes does not increase linearly while increasing N expect for Bernoulli model. The table below show the number of parameters with 4 typical canonical model form:

Potential	Coefficients number L	$\mathcal{O}(f(N, L))$
Bernoulli	N	$\mathcal{O}(N, N)$
Ising	$N + \frac{N(N-1)}{2}$	$\mathcal{O}(N, \frac{1}{2}(N + N^2))$
Triplets	$N + \frac{N(N-1)}{2} + \frac{N(N-1)(N-2)}{2}$	$\mathcal{O}(N, N + \frac{N^3}{6})$
Pairwise R	$N + \frac{N(N-1)(2R-1)}{2}$	$\mathcal{O}(N, N(R + \frac{3}{2}) + N^2(R - \frac{1}{2}))$

- **The time to update** $e^{\Delta\mathcal{H}}$ **is not linear in** L: in fact, while updating the Montecarlo spike train and computing $e^{\Delta\mathcal{H}}$ at each iteration, we compute it while making a sum over all the monomials and adding the change due to each of the monomials. Computing the changes due to each monomial is not the same for all the monomials. The reason is that some monomials contain 1 events (those of rates), some contain 2 events (pairwise) and others contain 3 events (triplets) and we might have more.

Those two facts explains why the curve shown about the computing time using Montecarlo is a power law in the linear scale (equivalent to a linear law in the logscale as in Figure 3.9).

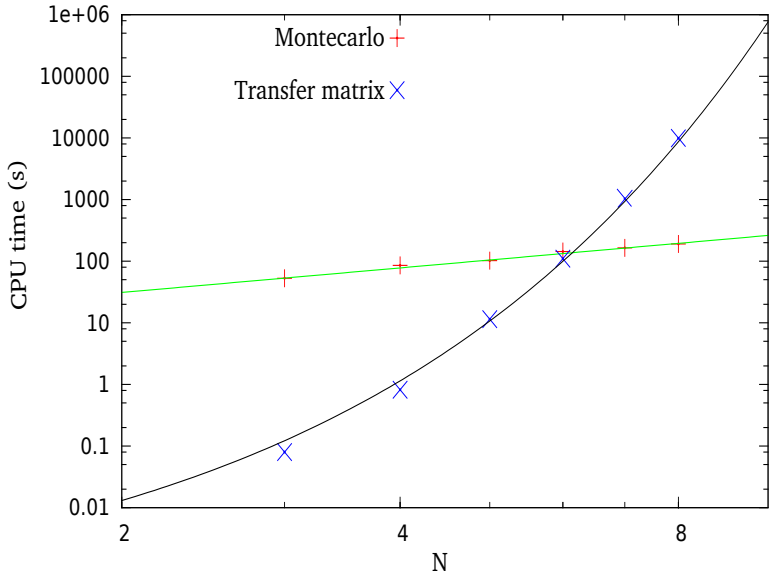


Figure 3.9: During the experiment of Figure 3.7, we computed the time needed to compute the target distribution with both methods. The +/× signs shows the computation time and the lines show the fit. In the case of Montecarlo, the computing time increases linearly with the number of neurons. However, with the transfer matrix, the time increases exponentially.

3.5.1 Using the Montecarlo principle to compute the Kullback-Leibler divergence

As for large networks we cannot compute the exact value of pressure and monomials averages, then having an exact value of the KLD is impossible. In this case, we compute and estimated divergence. From Eq. 2.44, we can write:

$$\begin{aligned} D_{kl}^{(E)}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}}) &= \mu_{\boldsymbol{\lambda}}[\mathcal{H}] + \mathcal{S}[\mu_{\boldsymbol{\lambda}}] - \pi_\omega^{(T)}[\mathcal{H}] - \mathcal{S}\left[\pi_\omega^{(T)}\right] \\ &= \mu_{\boldsymbol{\lambda}}[\mathcal{H}] - \pi_\omega^{(T)}[\mathcal{H}] + \mathcal{S}[\mu_{\boldsymbol{\lambda}}] - \mathcal{S}\left[\pi_\omega^{(T)}\right] \\ &= \sum_l \lambda_l \big(\mu_{\boldsymbol{\lambda}}[m_l] - \pi_\omega^{(T)}[m_l]\big) + \mathcal{S}[\mu_{\boldsymbol{\lambda}}] - \mathcal{S}\left[\pi_\omega^{(T)}\right] \end{aligned} \tag{3.8}$$

From the parameters $\boldsymbol{\lambda}$, we compute a spike train distributed as $\boldsymbol{\lambda}$ using the Montecarlo method. From this spike train, we compute the monomials of averages $\mu_{\boldsymbol{\lambda}}[m_l]$ and the entropy $\mathcal{S}[\mu_{\boldsymbol{\lambda}}]$ using the method of strong (Strong et al., 1998). $\pi_\omega^{(T)}[m_l]$ and $\mathcal{S}[\pi_\omega^{(T)}]$ are computed directly from the

empirical data.

3.6 Fitting the parameters λs

Eq. 2.20 or 2.38 provide an analytical way to compute the coefficients of the Gibbs distribution from data. However, they require the computation of the partition function or of the topological pressure which becomes rapidly intractable as the number of neurons increases. Thus, researchers have attempted to find alternative methods to compute reliably and efficiently the λ_ls. An efficient method has been introduced in [Dudík et al., 2004] and applied to spike trains in [Broderick et al., 2007]. Although these papers are restricted to Gibbs distributions of the form (2.18) (models without memory) we show in this section how their method can be extended to general Gibbs distributions.

3.6.1 Bounding the Kullback-Leibler divergence variation

The spatial case

The method developed in [Dudík et al., 2004] by Dudik et al is based on the so-called convex duality principle, used in mathematical optimization theory. Due the difficulty in maximizing the entropy (which is a concave function), one looks for a convex function easier to investigate. Dudik et al showed that, for spatially constrained Maxent distributions, finding the Gibbs distribution amounts to finding the minimum of the negative log likelihood[2]:

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = -\pi_\omega^{(T)}\left[\log \mu_{\boldsymbol{\lambda}}\right]. \tag{3.9}$$

Indeed, in the spatial case, the Kullback-Leibler divergence between the empirical measure $\pi_\omega^{(T)}$ and the Gibbs distribution at $\mu_{\boldsymbol{\lambda}}$ is:

$$d_{KL}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}}) = \pi_\omega^{(T)}\left[\frac{\log \pi_\omega^{(T)}}{\log \mu_{\boldsymbol{\lambda}}}\right] = \pi_\omega^{(T)}\left[\log \pi_\omega^{(T)}\right] - \pi_\omega^{(T)}\left[\log \mu_{\boldsymbol{\lambda}}\right], \tag{3.10}$$

so that, from Eq. 2.43:

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = \mathcal{P}\left[\boldsymbol{\lambda}\right] - \pi_\omega^{(T)}\left[\mathcal{H}\right]$$

[2]We have adapted [Dudík et al., 2004] to our notations. Moreover, in our case $\pi_\omega^{(T)}$ corresponds to the empirical average on a raster ω whereas π in [Dudík et al., 2004] corresponds to an average over independent samples.

Since $\mathcal{P}$ is convex and $\pi_\omega^{(T)}[\mathcal{H}]$ linear in $\boldsymbol{\lambda}$, $L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda})$ is convex. Its unique minimum is given by Eq. 2.20.

Moreover, we have:

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = \mathcal{P}[\boldsymbol{\lambda}'] - \mathcal{P}[\boldsymbol{\lambda}] - \pi_\omega^{(T)}[\Delta\mathcal{H}], \tag{3.11}$$

with $\Delta\mathcal{H} = \mathcal{H}_{\boldsymbol{\lambda}'} - \mathcal{H}$. From Eq. 2.18:

$$\begin{aligned}\frac{Z[\boldsymbol{\lambda}']}{Z[\boldsymbol{\lambda}]} &= \frac{1}{Z[\boldsymbol{\lambda}]}\sum_{\omega(0)} e^{\mathcal{H}_{\boldsymbol{\lambda}'}(\omega(0))} \\ &= \sum_{\omega(0)} e^{\Delta\mathcal{H}(\omega(0))}\mu_{\boldsymbol{\lambda}}[\omega(0))] \\ &= \mu_{\boldsymbol{\lambda}}\left[e^{\Delta\mathcal{H}_{\boldsymbol{\lambda}}}\right] \end{aligned} \tag{3.12}$$

and since $P[\boldsymbol{\lambda}] = \log Z[\boldsymbol{\lambda}]$ in the spatial case:

$$\mathcal{P}[\boldsymbol{\lambda}'] - \mathcal{P}[\boldsymbol{\lambda}] = \log \mu_{\boldsymbol{\lambda}}\left[e^{\Delta\mathcal{H}}\right]. \tag{3.13}$$

Therefore:

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = \log \mu_{\boldsymbol{\lambda}}\left[e^{\Delta\mathcal{H}}\right] - \pi_\omega^{(T)}[\Delta\mathcal{H}]. \tag{3.14}$$

The idea proposed by Dudik et al is then to bound this difference by an easier-to-compute convex quantity, with the same minimum as $L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda})$, and to reach this minimum by iterations on $\boldsymbol{\lambda}$. They proposed a sequential and a parallel method. Let us summarize first the sequential method. The goal here is not to rewrite their paper [Dudík et al., 2004] but to explain some crucial elements that are not directly appliable to the spatio-temporal case.

In the sequential case one updates $\boldsymbol{\lambda}$ as $\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \delta\boldsymbol{e}_l$, for some l, where $\boldsymbol{e}_l$ is the canonical basis vector in direction l, so that $\Delta\mathcal{H} = \delta m_l$, and

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = \log \mu_{\boldsymbol{\lambda}}\left[e^{\delta m_l}\right] - \delta\pi_\omega^{(T)}[m_l].$$

Using the following property:

$$e^{\delta x} \leq 1 + (e^\delta - 1)x, \tag{3.15}$$

for $x \in [0,1]$ and since $m_l \in \{0,1\}$, we have:

$$\log \mu_{\boldsymbol{\lambda}}\left[e^{\delta m_l}\right] \leq \log\left(1 + (e^\delta - 1)\mu_{\boldsymbol{\lambda}}[m_l]\right). \tag{3.16}$$

This bound, proposed by Dudik et al, is remarkably clever. Indeed, it replaces the computation of the average $\mu_{\boldsymbol{\lambda}}\left[\, e^{\delta m_l}\,\right]$, which is computationally hard, by the computation of $\mu_{\boldsymbol{\lambda}}\left[\, m_l\,\right]$, which is computationally easy. Finally,

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) \leq -\delta\pi_\omega^{(T)}\left[\, m_l\,\right] + \log\left(\, 1 + (e^\delta - 1)\mu_{\boldsymbol{\lambda}}\left[\, m_l\,\right]\right). \quad (3.17)$$

In the parallel case, the computation and results differ. One now updates $\boldsymbol{\lambda}$ as $\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \sum_{l=1}^{L}\delta_l \boldsymbol{e}_l$. Moreover, one has to renormalize the m_ls in $m'_l = \frac{m_l}{L}$ in order that Eq. 3.18 below holds. We have therefore $\Delta\mathcal{H} = \sum_{l=1}^{L}\delta_l m'_l$.

Thus,

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) = \log\mu_{\boldsymbol{\lambda}}\left[\, e^{\sum_{l=1}^{L}\delta_l m'_l}\,\right] - \sum_{l=1}^{L}\delta_l\pi_\omega^{(T)}\left[\, m'_l\,\right].$$

Using the following property [Collins et al., 2002]:

$$e^{\sum_{l=1}^{L}\delta_l m'_l} \leq 1 + \sum_{l=1}^{L} m'_l\,\left(\, e^{\delta_l} - 1\,\right), \quad (3.18)$$

for $\delta_l \in \mathbb{R}$ and $m'_l \geq 0$, $\sum_{l=1}^{L} m'_l \leq 1$, we have:

$$\log\mu_{\boldsymbol{\lambda}}\left[\, e^{\sum_{l=1}^{L}\delta_l m'_l}\,\right] \leq \log\left(\, 1 + \sum_{l=1}^{L}\left(\, e^{\delta_l} - 1\,\right)\,\mu_{\boldsymbol{\lambda}}[m'_l]\,\right).$$

Since $\log(1+x) \leq x$ for $x > -1$, Dudick et al obtain that:

$$\log\mu_{\boldsymbol{\lambda}}\left[\, e^{\sum_{l=1}^{L}\delta_l m'_l}\,\right] \leq \sum_{l=1}^{L}\left(\, e^{\delta_l} - 1\,\right)\,\mu_{\boldsymbol{\lambda}}[m'_l],$$

provided $\sum_{l=1}^{L}\left(\, e^{\delta_l} - 1\,\right)\,\mu_{\boldsymbol{\lambda}}[m'_l] > -1$. (This constraint has to be checked during iterations). Finally, using the definition of m'_l:

$$L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}') - L_{\pi_\omega^{(T)}}(\boldsymbol{\lambda}) \leq \frac{1}{L}\left[\, -\sum_{l=1}^{L}\delta_l\pi_\omega^{(T)}\left[\, m_l\,\right] + \sum_{l=1}^{L}\left(\, e^{\delta_l} - 1\,\right)\,\mu_{\boldsymbol{\lambda}}[m_l]\,\right]. \quad (3.19)$$

To be complete, let us mention that Dudik et al consider the case where some error ϵ_l is allowed in the estimation of the coefficient λ_l. This relaxation on the parameters alleviates the overfitting.

In this case, the bound on the right hand side in Eq. 3.17 (sequential case) becomes:

$$F_l(\boldsymbol{\lambda}, \delta) = -\delta \pi_\omega^{(T)}[\, m_l \,] + \log\left(1 + (e^\delta - 1)\mu_{\boldsymbol{\lambda}}[\, m_l \,]\right) + \epsilon_l\left(|\, \lambda_l + \delta \,| - |\, \lambda_l \,|\right). \tag{3.20}$$

whereas the right hand side in Eq. 3.19 becomes $\sum_{l=1}^{L} G_l(\boldsymbol{\lambda}, \boldsymbol{\delta})$ with:

$$G_l(\boldsymbol{\lambda}, \boldsymbol{\delta}) = \frac{1}{L}\left[-\delta_l \pi_\omega^{(T)}[\, m_l \,] + \left(e^{\delta_l} - 1\right)\mu_{\boldsymbol{\lambda}}[m_l]\right] + \epsilon_l\left(|\, \lambda_l + \delta \,| - |\, \lambda_l \,|\right), \tag{3.21}$$

The minimum of these functions is easy to find and one obtains, for a given $\boldsymbol{\lambda}$ the variation $\boldsymbol{\delta}$ required to lower bound the log-likelihood variation. The authors have shown that both sequential and parallel method produce a sequence $\boldsymbol{\lambda}^{(k)}$ which converges to the minimum of $L_{\pi_\omega^{(T)}}$ as $k \to +\infty$.

Extension to the spatio-temporal case

We now show how to extend these computations to the spatio-temporal case, provided one replaces the log-likelihood $L_{\pi_\omega^{(T)}}$ by the Kullback-Leibler divergence (Eq. 2.43). The main obstacle is that the Gibbs distribution does not have the form $\frac{e^{\mathcal{H}}}{Z}$. We obtain thus a convex criterion to minimize Kullback-Leibler divergence variation, hence reaching it minimum, $\pi_\omega^{(T)}$.

Replacing ν in Eq. (2.43) by $\pi_\omega^{(T)}$, the empirical measure, one has:

$$d_{KL}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}'}) - d_{KL}(\pi_\omega^{(T)}, \mu_{\boldsymbol{\lambda}}) = \mathcal{P}[\, \boldsymbol{\lambda}' \,] - \mathcal{P}[\, \boldsymbol{\lambda} \,] - \pi_\omega^{(T)}[\, \Delta\mathcal{H} \,], \tag{3.22}$$

because the entropy $\mathcal{S}\left[\pi_\omega^{(T)}\right]$ cancels. This is the analogue of Eq. (3.11). The main problem now is to compute $\mathcal{P}[\, \boldsymbol{\lambda}' \,] - \mathcal{P}[\, \boldsymbol{\lambda} \,]$.

Gibbs distribution obeys (Bowen, 2008):

$$A \leq \frac{\mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right]}{e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} e^{\mathcal{H}(\omega_0^{n-1})}} \leq B, \tag{3.23}$$

Then,

$$\begin{aligned}
& A e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} e^{\mathcal{H}(\omega_0^{n-1})} e^{\Delta\mathcal{H}(\omega_0^{n-1})} \\
& \leq \sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right] e^{\Delta\mathcal{H}(\omega_0^{n-1})} \\
& \leq B e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} e^{\mathcal{H}(\omega_0^{n-1})} e^{\Delta\mathcal{H}(\omega_0^{n-1})}
\end{aligned}$$

so that:

$$\begin{aligned}
&\lim_{n\to\infty}\frac{1}{n}\left[\log A-(n-D)\mathcal{P}\left[\boldsymbol{\lambda}\right]+\log\left(\sum_{\omega_0^{n-1}}e^{\mathcal{H}(\omega_0^{n-1})}e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right)\right]\\
&\leq\lim_{n\to\infty}\frac{1}{n}\log\left(\sum_{\omega_0^{n-1}}\mu_{\boldsymbol{\lambda}}\left[\,\omega_0^{n-1}\,\right]e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right)\\
&\leq\lim_{n\to\infty}\frac{1}{n}\left[\log B-(n-D)\mathcal{P}\left[\boldsymbol{\lambda}\right]+\log\left(\sum_{\omega_0^{n-1}}e^{\mathcal{H}(\omega_0^{n-1})}e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right)\right].
\end{aligned}\tag{3.24}$$

Since $\mathcal{H}_{\boldsymbol{\lambda}'}(\omega_0^{n-1})=\mathcal{H}(\omega_0^{n-1})+\Delta\mathcal{H}(\omega_0^{n-1})$, from Eq. 2.34:

$$\lim_{n\to\infty}\frac{1}{n}\log\sum_{\omega_0^{n-1}}e^{\mathcal{H}(\omega_0^{n-1})}e^{\Delta\mathcal{H}(\omega_0^{n-1})}=\mathcal{P}\left[\boldsymbol{\lambda}'\right].$$

Therefore:

$$\mathcal{P}\left[\boldsymbol{\lambda}'\right]-\mathcal{P}\left[\boldsymbol{\lambda}\right]=\lim_{n\to\infty}\frac{1}{n}\log\sum_{\omega_0^{n-1}}\mu_{\boldsymbol{\lambda}}\left[\,\omega_0^{n-1}\,\right]e^{\Delta\mathcal{H}(\omega_0^{n-1})}.\tag{3.25}$$

This is the extension of Eq. 3.13 to the spatio temporal case. In the spatial case it reduces to Eq. 3.13 from Eq. 2.13. This equation is obviously numerically intractable, but it has two advantages: on one hand it allows to extend the bounds (Eq. 3.17, sequential case) and (Eq. 3.19, parallel case), and on the other hand it can be used to get a $\boldsymbol{\delta}$-power expansion of $\mathcal{P}\left[\boldsymbol{\lambda}'\right]-\mathcal{P}\left[\boldsymbol{\lambda}\right]$. This last point is used in the section 3.6.2.

To get the analogue of Eq. 3.17 in the sequential case where $\Delta\mathcal{H}(\omega_0^{n-1})=\delta\sum_{r=0}^{n-D-1}m_l(\omega_r^{r+D})$, one may still apply Eq. 3.15 which holds provided:

$$m_l(\omega_0^{n-1})\equiv\sum_{r=0}^{n-1-D}m_l(\omega_r^{r+D})<1\tag{3.26}$$

So, compared to the spatial we have to replace m_l by $\frac{m_l}{n-D}$ in $\Delta\mathcal{H}(\omega_0^{n-1})$. We have therefore:

$$\begin{aligned}\sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}[\omega_0^{n-1}] e^{\Delta\mathcal{H}(\omega_0^{n-1})} &= \sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}\left[\,\omega_0^{n-1}\,\right] e^{\delta\frac{1}{n-D}m_l(\omega_0^{n-1})} \\ &\leq 1+(e^{\delta}-1)\frac{1}{n-D}\sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}\left[\,\omega_0^{n-1}\,\right] m_l(\omega_0^{n-1}).\end{aligned}$$

From the time translation invariance of $\mu_{\boldsymbol{\lambda}}$ we have:

$$\begin{aligned}\frac{1}{n-D}\sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}[\omega_0^{n-1}] m_l(\omega_0^{n-1}) &= \frac{1}{n-D}\sum_{r=0}^{n-D-1}\sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}[\omega_0^{n-1}] m_l(\omega_r^{r+D}) \\ &= \frac{1}{n-D}\sum_{r=0}^{n-D-1} \mu_{\boldsymbol{\lambda}}[m_l] \\ &= \mu_{\boldsymbol{\lambda}}[m_l]\end{aligned}$$

so that:

$$\sum_{\omega_0^{n-1}} \mu_{\boldsymbol{\lambda}}\left[\,\omega_0^{n-1}\,\right] e^{\delta\frac{1}{n-D}m_l(\omega_0^{n-1})} \leq 1+(e^{\delta}-1)\mu_{\boldsymbol{\lambda}}\left[\,m_l\,\right].$$

At first glance this bound is not really useful. Indeed, from Eq. 3.25 we obtain:

$$\mathcal{P}\left[\,\boldsymbol{\lambda}'\,\right]-\mathcal{P}\left[\,\boldsymbol{\lambda}\,\right] \leq \lim_{n\to\infty}\frac{1}{n}\log\left(\,1+(e^{\delta}-1)\mu_{\boldsymbol{\lambda}}\left[\,m_l\,\right]\right)=0.$$

Since this holds for any δ this implies $\mathcal{P}\left[\,\boldsymbol{\lambda}'\,\right]=\mathcal{P}\left[\,\boldsymbol{\lambda}\,\right]$. The reason for this is evident. Renormalizing m_l as we did to match the condition imposed by bound (Eq. 3.15) is equivalent to renormalizing δ by $\frac{\delta}{n-D}$. As $n\to+\infty$ this perturbation tends to 0 and $\boldsymbol{\lambda}'=\boldsymbol{\lambda}$. Therefore, the clever bound (Eq. 3.15) would be of no interest if we were seeking exact results. However, the goal here is to propose a numerical scheme, where, obviously n is finite. We replace therefore the limit $n\to\infty$ by a fixed n in the computation of $\mathcal{P}\left[\,\boldsymbol{\lambda}'\,\right]-\mathcal{P}\left[\,\boldsymbol{\lambda}\,\right]$. Keeping in mind that m_l must also be renormalized in $\pi_{\omega}^{(T)}\left[\,\Delta\mathcal{H}\,\right]$ and using $\frac{1}{n}<\frac{1}{n-D}$ the Kullback-Leibler difference (Eq. 3.22) obeys:

$$d_{KL}(\pi_{\omega}^{(T)},\mu_{\boldsymbol{\lambda}'})-d_{KL}(\pi_{\omega}^{(T)},\mu_{\boldsymbol{\lambda}}) \leq \frac{1}{n-D}\left[\,-\delta\pi_{\omega}^{(T)}\left[\,m_l\,\right]+\log\left(\,1+(e^{\delta}-1)\mu_{\boldsymbol{\lambda}}\left[\,m_l\,\right]\right)\right], \tag{3.27}$$

the analog of Eq. 3.17.

In the parallel case, similar remarks holds. In order to apply the bound (Eq. 3.18) we have to renormalize the m_ls in $m_l'=\frac{1}{L(n-D)}$. As for the

spatial case we also need to check that $\sum_{l=1}^{L}\left(e^{\delta_l}-1\right)\mu_{\boldsymbol{\lambda}}[m'_l] > -1$. (This constraint is not guarantee and has to be checked during iterations). One obtains finally:

$$d_{KL}(\pi_\omega^{(T)},\mu_{\boldsymbol{\lambda}'})-d_{KL}(\pi_\omega^{(T)},\mu_{\boldsymbol{\lambda}}) \leq \frac{1}{L(n-D)}\left[-\sum_{l=1}^{L}\delta_l\pi_\omega^{(T)}\left[\,m_l\,\right]+\sum_{l=1}^{L}\left(e^{\delta_l}-1\right)\mu_{\boldsymbol{\lambda}}[m_l]\right], \tag{3.28}$$

the analogue of Eq. 3.19.

Compared with the spatial case, we see therefore that n mustn't be too large to have a reasonable Kullback-Leibler divergence variation. It mustn't be too small, however, to get a good approximation of the topological pressure.

3.6.2 Updating the target distribution when the parameters change

When updating the parameters $\boldsymbol{\lambda}$, one has to compute again the average values $\mu_{\boldsymbol{\lambda}}\left[\,m_l\,\right]$ since the probability $\mu_{\boldsymbol{\lambda}}$ has changed. This has a huge computational cost. The exact computation (e.g. from Eq. 2.20 and 2.38) is not tractable for large N so approximate methods have to be used, like Montecarlo (Nasser et al., 2013). Again, this is also CPU time consuming especially if one recomputes it again at each iteration, but at least it is tractable.

In this spirit, Broderick et al Broderick et al., 2007 propose to generate a Montecarlo raster distributed according to $\mu_{\boldsymbol{\lambda}}$ and to use it to compute $\mu_{\boldsymbol{\lambda}'}$ when $\|\boldsymbol{\lambda}'-\boldsymbol{\lambda}\|$ is sufficiently small. We expose their method, limited to the spatial case, in the next section, and we explain why it is not applicable in the spatio-temporal case. We then propose an alternative method.

The spatial case

The average of m_l is obtained by the derivative of the topological pressure $\mathcal{P}[\boldsymbol{\lambda}]$. In the spatial case, where $\mathcal{P}(\boldsymbol{\lambda}) = \log Z_{\boldsymbol{\lambda}}$, we have:

$$\begin{aligned}\mu_{\boldsymbol{\lambda}'}[m_l] &= \frac{\partial \mathcal{P}(\boldsymbol{\lambda}')}{\partial \lambda'_j} \\ &= \frac{1}{Z[\boldsymbol{\lambda}']} \sum_{\omega(0)} m_l(\omega(0)) e^{\mathcal{H}_{\boldsymbol{\lambda}'}(\omega(0))} \\ &= \frac{Z[\boldsymbol{\lambda}]}{Z[\boldsymbol{\lambda}']} \sum_{\omega(0)} m_l(\omega(0)) e^{\Delta\mathcal{H}(\omega(0))} \mu_{\boldsymbol{\lambda}}[\omega(0))] \end{aligned} \tag{3.29}$$

Using Eq. 3.12, one finally obtains:

$$\mu_{\boldsymbol{\lambda}'}[m_l] = \frac{\mu_{\boldsymbol{\lambda}}\left[m_l(\omega(0))\, e^{\Delta\mathcal{H}(\omega(0))}\right]}{\mu_{\boldsymbol{\lambda}}\left[e^{\Delta\mathcal{H}(\omega(0))}\right]}, \tag{3.30}$$

which is eq. (18) in [Broderick et al., 2007]. Using this formula one is able to compute the average of m_l with respect to the new probability $\mu_{\boldsymbol{\lambda}'}$ only using the old one, $\mu_{\boldsymbol{\lambda}}$.

Extension to the spatio-temporal case

We now explain why the Broderick et al method does not extend to the spatio-temporal case. The main problem is that if one tries to obtain the analogue of the Eq. 3.30 one obtains in fact an inequality:

$$\frac{A}{B}\mu_{\boldsymbol{\lambda}'}[m_l] \leq \lim_{n\to\infty} \frac{1}{n} \frac{\mu_{\boldsymbol{\lambda}}\left[m_l\left(\omega_0^{n-1}\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right]}{\mu_{\boldsymbol{\lambda}}\left[e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right]} \leq \frac{B}{A}\mu_{\boldsymbol{\lambda}'}[m_l], \tag{3.31}$$

where A, B are the constants in Eq. 3.23. They are not known in general (they depend on the potential) and they are different. However, in the spatial case $A = B = 1$ whereas $\mu_{\boldsymbol{\lambda}}\left[m_l\left(\omega_0^{n-1}\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})}\right] = \mu_{\boldsymbol{\lambda}}\left[m_l\left(\omega(0)\right) e^{\Delta\mathcal{H}(\omega(0))}\right]$ because the potential has range 1. Then, one recovers Eq. 3.30. Let us now explain how we obtain Eq. 3.31.

The averages of quantities are obtained by the derivative of the topological pressure (Eq. (2.38)). We have:

$$\mu_{\boldsymbol{\lambda}'}[m_l] = \frac{\partial \mathcal{P}}{\partial \lambda'_l} = \frac{\partial \lim_{n\to\infty} \frac{1}{n} \log Z_n[\boldsymbol{\lambda}']}{\partial \lambda'_l}. \tag{3.32}$$

Assuming that the limit and the derivative commute (see e.g. [Mayer and Urbański, 2010]), gives:

$$\begin{aligned}
\mu_{\boldsymbol{\lambda}'}\left[\, m_l \,\right] &= \lim_{n\to\infty} \frac{1}{n} \frac{1}{Z_n\left[\, \boldsymbol{\lambda}' \,\right]} \sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\mathcal{H}_{\boldsymbol{\lambda}'}(\omega_0^{n-1})} \\
&= \lim_{n\to\infty} \frac{1}{n} \frac{1}{Z_n\left[\, \boldsymbol{\lambda}' \,\right]} \sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})} \\
&= \lim_{n\to\infty} \frac{1}{n} \frac{\sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}}{\sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}} \\
&\quad . \qquad (3.33)
\end{aligned}$$

From Eq. 3.23:

$$\begin{aligned}
& A\, e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})} \\
& \leq \sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} \mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right] \\
& \leq B\, e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}
\end{aligned} \qquad (3.34)$$

and:

$$\begin{aligned}
& A\, e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})} \\
& \leq \sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} \mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right] \\
& \leq B\, e^{-(n-D)\mathcal{P}[\boldsymbol{\lambda}]} \sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}.
\end{aligned}$$

Therefore:

$$\begin{aligned}
& \frac{A}{B} \frac{\sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}}{\sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}} \\
& \leq \frac{\sum_{\omega_0^{n-1}} m_l\left(\, \omega_0^{n-1} \,\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} \mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right]}{\sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} \mu_{\boldsymbol{\lambda}}\left[\, \omega_0^{n-1} \,\right]}
\end{aligned}$$

$$\leq \frac{B}{A}\frac{\sum_{\omega_0^{n-1}} m_l\left(\omega_0^{n-1}\right) e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}}{\sum_{\omega_0^{n-1}} e^{\Delta\mathcal{H}(\omega_0^{n-1})} e^{\mathcal{H}(\omega_0^{n-1})}}.$$

Now, from [Chazottes and Keller, 2008, Keller, 1998], Eq. 3.33) gives 3.31.

Taylor expansion of the pressure

The idea is here to use a Taylor expansion of the topological pressure. This approach is very much in the spirit of [Kappen and Rodriguez, 1998], but extended here to the spatio-temporal case. Since $\boldsymbol{\lambda}' = \boldsymbol{\lambda} + \boldsymbol{\delta}$, we have:

$$\begin{aligned}\mu_{\boldsymbol{\lambda}'}[\,m_l\,] &= \mu_{\boldsymbol{\lambda}}[\,m_l\,] + \sum_{j=1}^{L}\frac{\partial\mu_{\boldsymbol{\lambda}}[\,m_l\,]}{\partial\lambda_j}\delta_j + \frac{1}{2}\sum_{j,k=1}^{L}\frac{\partial^2\mu_{\boldsymbol{\lambda}}[\,m_l\,]}{\partial\lambda_j\partial\lambda_k}\delta_j\delta_k + \ldots \\ &= \mu_{\boldsymbol{\lambda}}[\,m_l\,] + \sum_{j=1}^{L}\frac{\partial^2\mathcal{P}[\,\boldsymbol{\lambda}\,]}{\partial\lambda_j\partial\lambda_l}\delta_j + \frac{1}{2}\sum_{j,k=1}^{L}\frac{\partial^3\mathcal{P}[\,\boldsymbol{\lambda}\,]}{\partial\lambda_j\partial\lambda_k\partial\lambda_l}\delta_j\delta_k + . \quad (3.35)\end{aligned}$$

The second derivative of the pressure is given by [Ruelle, 1978, Bowen, 1975, Georgii, 1988, Chazottes and Keller, 2008]:

$$\frac{\partial^2\mathcal{P}[\,\boldsymbol{\lambda}\,]}{\partial\lambda_j\partial\lambda_l} = \sum_{n=-\infty}^{+\infty} C_{jl}(n) \equiv \chi_{jl}[\,\boldsymbol{\lambda}\,], \quad (3.36)$$

where:

$$C_{jl}(n) = \mu_{\boldsymbol{\lambda}}[\,m_j\, m_l\circ\sigma^n\,] - \mu_{\boldsymbol{\lambda}}[\,m_j\,]\,\mu_{\boldsymbol{\lambda}}[\,m_l\,], \quad (3.37)$$

is the correlation function between m_l, m_k at time n, computed with respect to $\mu_{\boldsymbol{\lambda}}$. (3.36) is a version of the fluctuation-dissipation theorem in the spatio-temporal case. σ^n is the time shift applied n times. The third derivatives can be computed as well by taking the derivative (3.36) and using (3.32). This generates terms with third order correlations and so on [Mayer and Urbański, 2010]. Up to second order we have:

$$\mu_{\boldsymbol{\lambda}'}[\,m_l\,] = \mu_{\boldsymbol{\lambda}}[\,m_l\,] + \sum_{j=1}^{L}\chi_{jl}[\,\boldsymbol{\lambda}\,]\,\delta_j + \ldots \quad (3.38)$$

Since the observable are monomials they only take the values 0 or 1 and the computation of χ_{jl} is straightforward, reducing to counting the occurrence of time pairs $t, t+n$ such that $m_j(t) = 1$ and $m_l(t+n) = 1$.

On practical grounds we introduce a parameter $\Delta = \|\boldsymbol{\lambda}' - \boldsymbol{\lambda}\|$ which measures the variation in the parameters after update. If Δ is small enough

(smaller than some Δ_c), the terms of order 3 in the Tayor expansion are negligible, then we can use (3.38). Otherwise, if Δ is big, we compute a new Montecarlo estimation of $\mu'_{\boldsymbol{\lambda}}$ (as described in [Nasser et al., 2013]). We explain in section 3.7.2 how Δ_c was chosen in our data. Then, we use the following trick. If $\|\boldsymbol{\delta}\| > \Delta_c$ we compute the new value $\mu_{\boldsymbol{\lambda}'}[m_j]$. If $\Delta_c > \|\boldsymbol{\delta}\| > \frac{\Delta_c}{10}$, we use the linear response approximation (3.38) of $\mu_{\boldsymbol{\lambda}'}$. Finally, if $\|\boldsymbol{\delta}\| < \frac{\Delta_c}{10}$ we use $\mu_{\boldsymbol{\lambda}}[m_l]$ instead of $\mu_{\boldsymbol{\lambda}'}[m_l]$ in the next iteration of the method . Thus, in the case, $\|\boldsymbol{\delta}\| < \Delta_c$, we use the Gibbs distribution computed at some time step, say n, to infer the values at the next iteration. If we do that several successive time steps the distance to the original value $\boldsymbol{\lambda}_n$ of the parameters increases. So we compute the norm $\|\boldsymbol{\lambda}_n - \boldsymbol{\lambda}_{n+k}\|$ at each time step k, and we do not compute a new raster until this norm is larger than Δ_c.

3.6.3 The algorithms

We have two algorithms, sequential and parallel, which are very similar to Dudik el al. Especially, the convergence of their algorithms, proved in their paper, extends to our case since it only depends on the shape of the cost functions (Eq. 3.20 and 3.21). We describe here the algorithms coming out from the presented mathematical framework, in a sequential and parallel version. We iterate the algorithms until the distance $\epsilon = d\left(\mu_{\boldsymbol{\lambda}}, \pi_\omega^{(T)}\right)$ is smaller than some ϵ_c. We use the Hellinger distance:

$$d\left(\mu_{\boldsymbol{\lambda}}, \pi_\omega^{(T)}\right) = \frac{1}{\sqrt{2}}\sqrt{\sum_{l=1}^{L}\left(\sqrt{\pi_\omega^{(T)}(m_l)} - \sqrt{\mu_{\boldsymbol{\lambda}}(m_l)}\right)^2} \tag{3.39}$$

Sequential and parallel algorithms are shown respectively in table 1 and 2.

3.6.4 Demo: inferring the Maximum Entropy distribution during the iterations

Here we show, on an example of 10 neurons with $R = 3$, how the parameters, the error and the predicted probability of monomials evolves from the initial condition until convergence. The example is shown in Figure 3.10 by showing the results at several steps in the updating algorithm.

Input: The features probabilities $\pi_{\omega}^{(T)}[\, m_l \,]$
Output: The vector of parameters $\boldsymbol{\lambda}$
initialization: $\lambda_l = 0$ for every l, $\Delta = 0$
while $\epsilon > \epsilon_c$ **do**
    $(\delta, l) = \arg\min_{l,\delta} F_l(\boldsymbol{\lambda}, \delta)$
    $\lambda_l \leftarrow \lambda_l + \delta$
    $\Delta \leftarrow \Delta + |\, \delta \,|$
    if $\Delta > \Delta_c$ **then**
        Compute a new Gibbs sample using Montecarlo method [Nasser et al., 2013]
    else
        Compute the new features probabilities using Taylor expansion (Eq. 3.38)
    end
end

Algorithm 1: Sequential algorithm. δ is the learning rate by which we change the value of a parameter λ_l. ϵ is the convergence criterion (Eq. 3.39)). Δ is the parameter allowing us to decide whether we update the parameters change by computing a new Gibbs sample or by the Taylor expansion. F_j is given by Eq. 3.20.

Input: features probabilities $\pi_{\omega}^{(T)}[\, m_l \,]$
Output: parameters λ_l
initialization: $\lambda_l = 0$ for every l, $\Delta = 0$
while $\epsilon > \epsilon_c$ **do**
    for $l \leftarrow 1$ **to** L **do**
        $\delta_l = \arg\min_{\delta} G_l(\boldsymbol{\lambda}, \boldsymbol{\delta})$
    end
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \boldsymbol{\delta}$
    $\Delta \leftarrow \sqrt{\Delta^2 + \sum_{l=1}^{L} \delta_l^2}$
    if $\Delta > \Delta_c$ **then**
        Compute a new Gibbs sample using Montecarlo method [Nasser et al., 2013]
    else
        Compute the new features probabilities using Taylor expansion (Eq. 3.38)
    end
end

Algorithm 2: The parallel algorithm. G_l is given by Eq. 3.21.

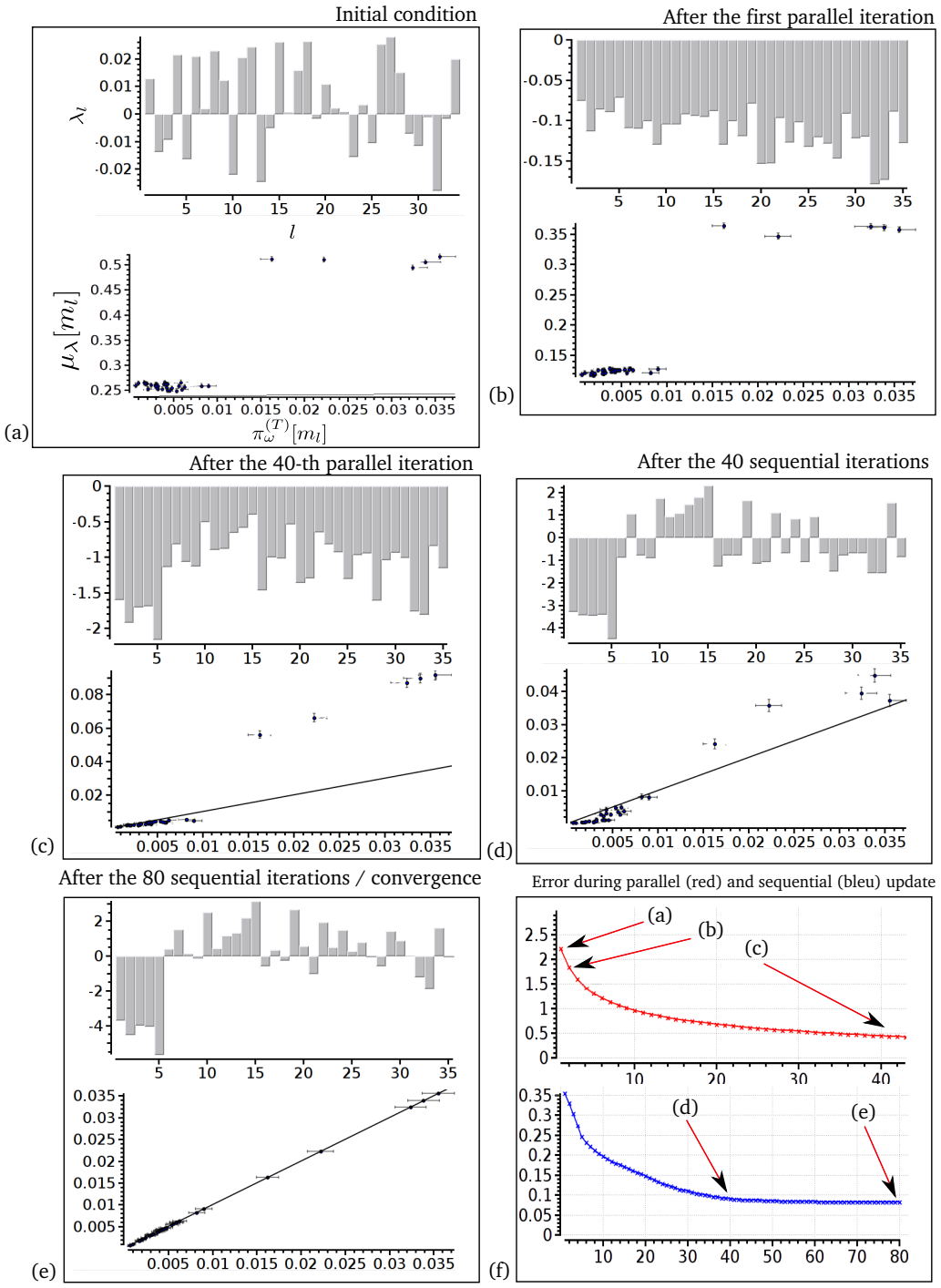


Figure 3.10: A demo for fitting the parameters. We adopted the strategy of iterating the parallel algorithm before the sequential one in order to have faster convergence. for small scale, 50 parallel iterations and 100 sequential are enough. For large scale, we need around 200 parallel iterations and as much parameters in the model for the sequential algorithm. (a)-(d) show the coefficients values and the comparison between expected averages and empirical averages of monomials (with errors bars) at some stages from the algorithm. (f) shows the Hellinger distance between the empirical and computed distribution during parallel and sequential update. The 5 neurons are chosen randomly from the data set (D_2) and binned with 20 ms.

3.7 Results on synthetic data

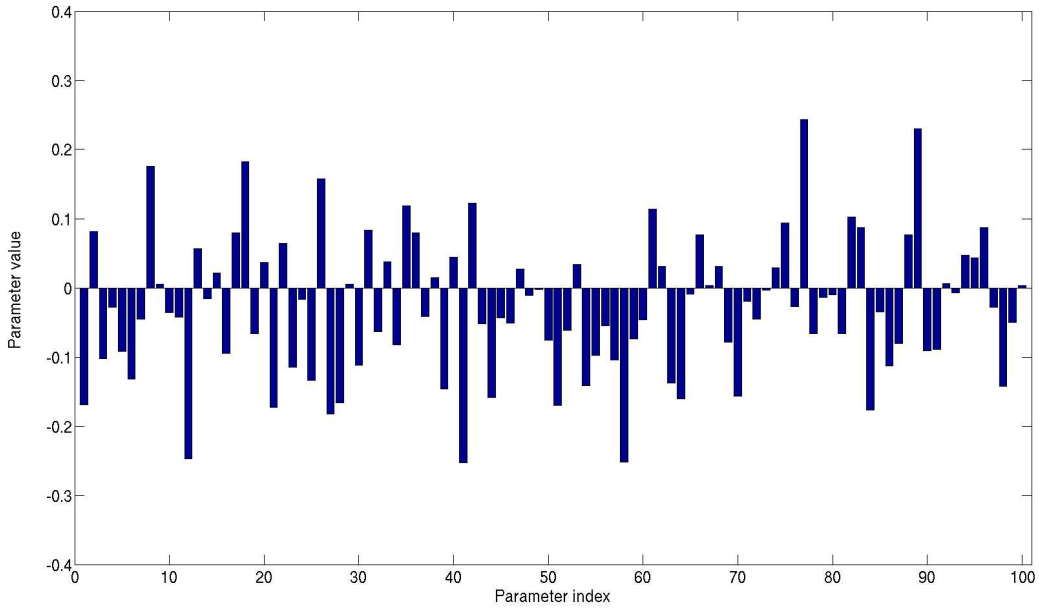
In this section we perform several tests on our method. We first consider synthetic data generated with a known Gibbs potential and recover its parameters. This step also allows us to tune the parameter Δ_c in the algorithms. Then, we consider real data analysis where the Gibbs potential form is unknown.

3.7.1 Synthetic data generation

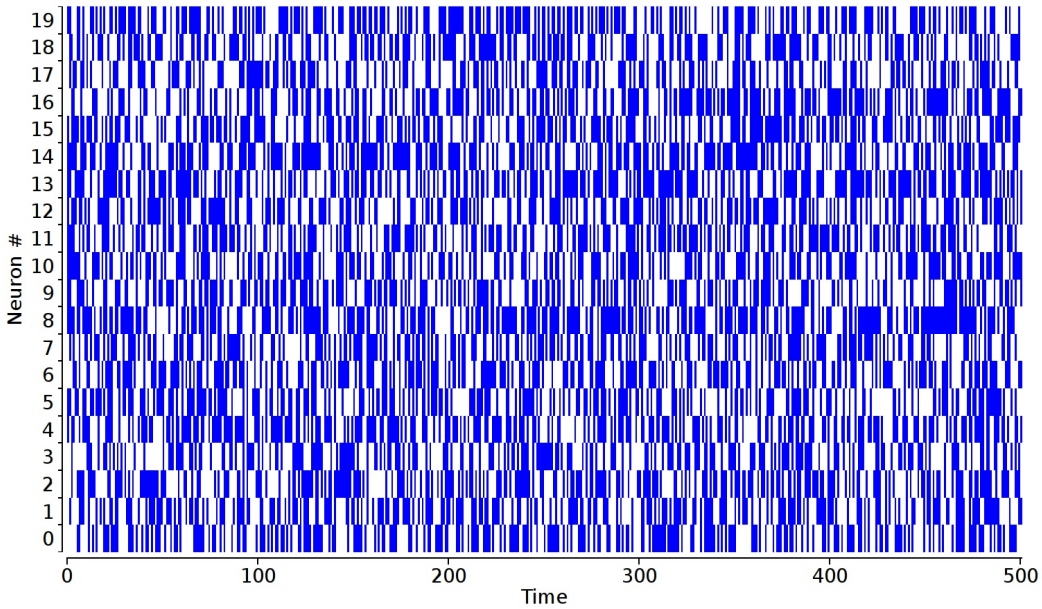
Synthetic data are obtained by generating a raster distributed according to a Gibbs distribution whose potential (2.4) is known. We consider two families of Gibbs potentials. For each family there are $L > N$ monomials whose range belongs to $\{1, \dots, R\}$. Among them, there are N "rate monomials" $\omega_i(D)$, $i = 1 \dots N$, whose average gives the firing rate of neuron i, denoted r_i ; the $L - N$ other monomials, with degree $k > 1$, are chosen at random with a probability law $\sim e^{-k}$ which favors therefore pairwise interactions. The difference between the two families comes from the distribution of coefficients λ_l.

1. **"Dense" rasters family**. The coefficients are drawn with a Gaussian distribution with mean 0 and variance $\frac{1}{L}$ to ensure a correct scaling of the coefficients dispersion as L increases (Figure 3.11(a)). This produces typically a dense raster (Figure 3.11(b)) with strong multiple correlations.

2. **"Sparse" rasters family.** The rate coefficients in the potential are very negative: the coefficient h_i of the rate monomial $\omega_i(D)$ is $h_i = \log\left(\frac{r_i}{1-r_i}\right)$ where $r_i \in [0 : 0.01]$ with a uniform probability distribution. Other coefficients are drawn with a Gaussian distribution with mean 0.8 and variance 1 (Figure 3.12(a)a). This produces a sparse raster (Figure 3.12(b)b) with strong multiple correlations.

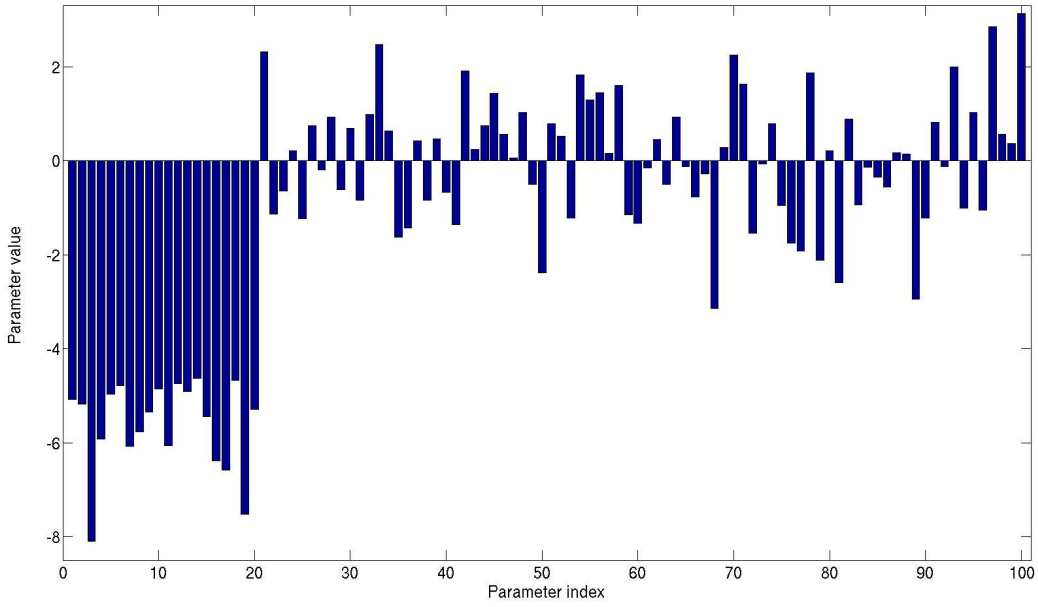


(a) Example of coefficients distribution in the dense rasters family: all the monomials coefficients are distributed as Gaussian.

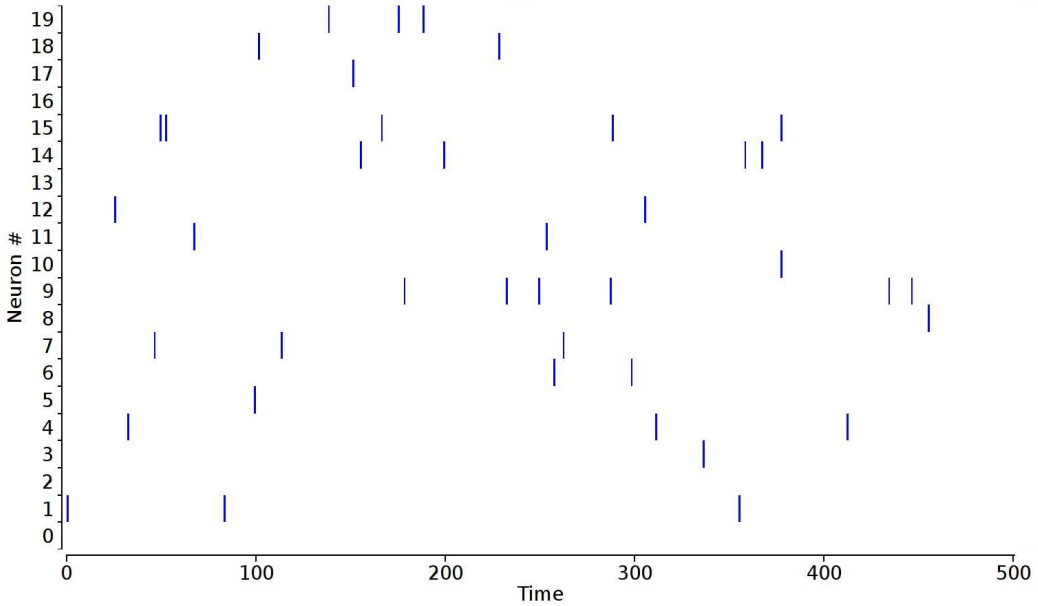


(b) Dense spike train

Figure 3.11: Dense spike train family.



(a) Example of coefficients distribution in the sparse rasters family: rates coefficients are negative and higher order monomials coefficients are distributed as Gaussian.



(b) Sparse spike train

Figure 3.12: Sparse spike train family.

3.7.2 Tuning Δ_c

For small N, R ($NR \leq 20$) it is possible to exactly compute the topological pressure using a transfer matrix technique [Vasquez et al., 2012]. We have therefore a way to compare the Taylor expansion (Eq. 3.36) and the exact value.

If we perturb $\boldsymbol{\lambda}$ by an amount δ in the direction j, this induces a variation on $\mu_{\boldsymbol{\lambda}}[m_j]$, $j = 1 \dots L$, given by the Taylor expansion (Eq. 3.38). To the lowest order $\mu_{\boldsymbol{\lambda}'}[m_j] = \mu_{\boldsymbol{\lambda}}[m_j] + O^{(1)}$, so that:

$$\epsilon^{(1)} = \frac{1}{L} \sum_{j=1}^{L} \frac{|\mu_{\boldsymbol{\lambda}'}[m_j] - \mu_{\boldsymbol{\lambda}}[m_j]|}{|\mu_{\boldsymbol{\lambda}'}[m_j]|}$$

is a measure of the relative error when considering the lowest order expansion.

In the same way, to the second order:

$$\mu_{\boldsymbol{\lambda}'}[m_j] = \mu_{\boldsymbol{\lambda}}[m_j] + \chi_{jk}[\boldsymbol{\lambda}]\,\delta_k + O^{(2)},$$

so that:

$$\epsilon^{(2)} = \frac{1}{L} \sum_{j=1}^{L} \frac{|\mu_{\boldsymbol{\lambda}'}[m_j] - \mu_{\boldsymbol{\lambda}}[m_j] - \chi_{jk}[\boldsymbol{\lambda}]\,\delta_k|}{|\mu_{\boldsymbol{\lambda}'}[m_j]|},$$

is a measure of the relative error when considering the next order expansion.

In Figure 3.13 we show the relative errors $\epsilon^{(1)}, \epsilon^{(2)}$ (in %), as a function of δ. For each point we generate 25 potentials, with $N = 5, R = 3, L = 12$. For each of these potentials we randomly perturb the λ_js, with a random sign, so that the norm of the perturbation $\|\boldsymbol{\delta}\|$ is fixed. The linear response χ is computed from a raster of length $T = 100000$.

These curves show a big difference between the dense and sparse case. In the dense case, the second order error is about 5% for $\Delta_c = 1$ whereas we need a $\Delta_c \sim 0.03$ to get the same 5% in the sparse case. We choose to align on the sparse case and in typical experiments we take $\Delta_c = 0.1$ corresponding to about 10% of error on the second order.

3.7.3 Test experiences

Here, we test the method on synthetic data where the shape of the sought potential is known: only the λ_ls have to be estimated. Experiments were designed according to the following steps:

- We start from a potential $\mathcal{H}_{\boldsymbol{\lambda}^*} = \sum_{l \in \mathcal{L}} \lambda_l^* m_l$. The goal is to estimate the coefficient values λ_l^* knowing the set $\mathcal{L}$ of monomials spanning the potential.

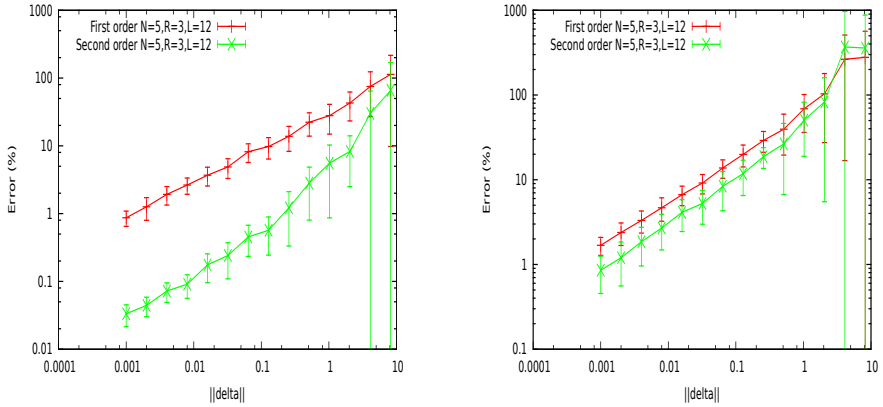
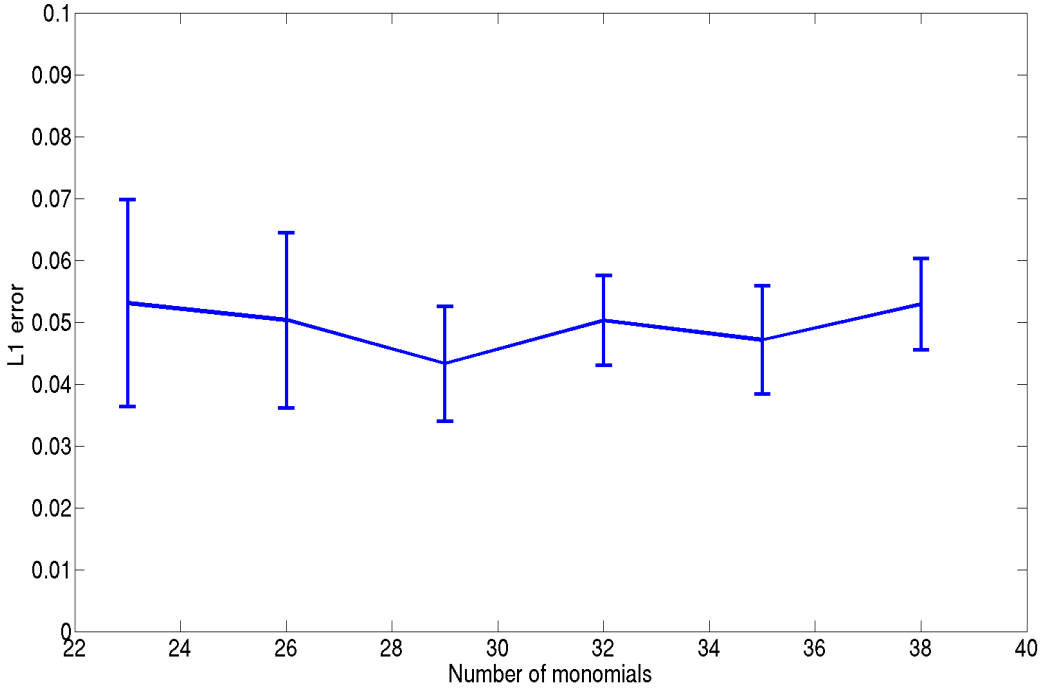


Figure 3.13: Error on the average $\mu_{\boldsymbol{\lambda}'}[\, m_j \,]$ as a function of the perturbation amplitude δ. First order corresponds to $\epsilon^{(1)}$ and second order to $\epsilon^{(2)}$ (see text). The curves correspond to $N = 5, R = 3, L = 12$. Left: Dense case; Right: Sparse case.

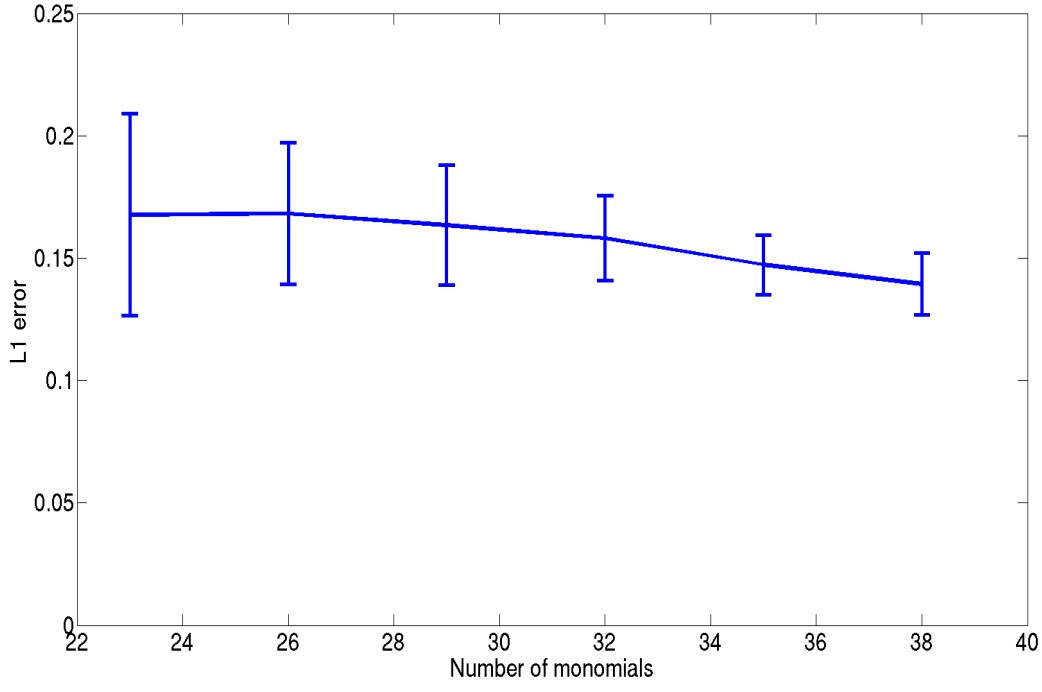
- We generate a synthetic spike train (ω_s) distributed according to the Gibbs distribution of $\mathcal{H}_{\boldsymbol{\lambda}^*}$.
- We take a potential $\mathcal{H}_{\boldsymbol{\lambda}} = \sum_{l \in \mathcal{L}} \lambda_l m_l$ with random initial coefficients λ_l. Then we fit the parameters λ_l to the synthetic spike train $\omega_s^{(T)}$.
- We evaluate the goodness of fit.

For the last step (goodness of fit) we have used two criteria. The first one simply consists of computing the L_1 error $d_1 = \frac{1}{L} \sum_{l=1}^{L} \left| \lambda_l^* - \lambda_l^{(est)} \right|$ where $\lambda_k^{(est)}$ is the final estimated value. d_1 is then averaged on 10 random potentials. Figure 3.14 shows the committed error in the case of sparse and dense potentials. The main advantage of this criterion is to provide an exact estimation of the error made on coefficients estimation. Its drawback is that we have to know the shape of the potential which generated the raster: this is not the case anymore for real neural networks data. The method showed a good performance, both in dense and sparse case, for large $N \times R \sim 60$.

We therefore used a second criterion (on sets of 40 neurons with Ising and pairwise models): confidence plots. For each spike block ω_0^D appearing in the raster ω_s we draw a point in a two dimensional diagram with, on abscissa, the observed empirical probability $\pi_{\omega_s}^{(T)}[\, \omega_0^D \,]$ and, on ordinate, the predicted probability $\mu_{\boldsymbol{\lambda}}[\, \omega_0^D \,]$. Ideally, all points should align on the diagonal $y = x$ (equality line). However, since the raster is finite there are finite-size



(a)



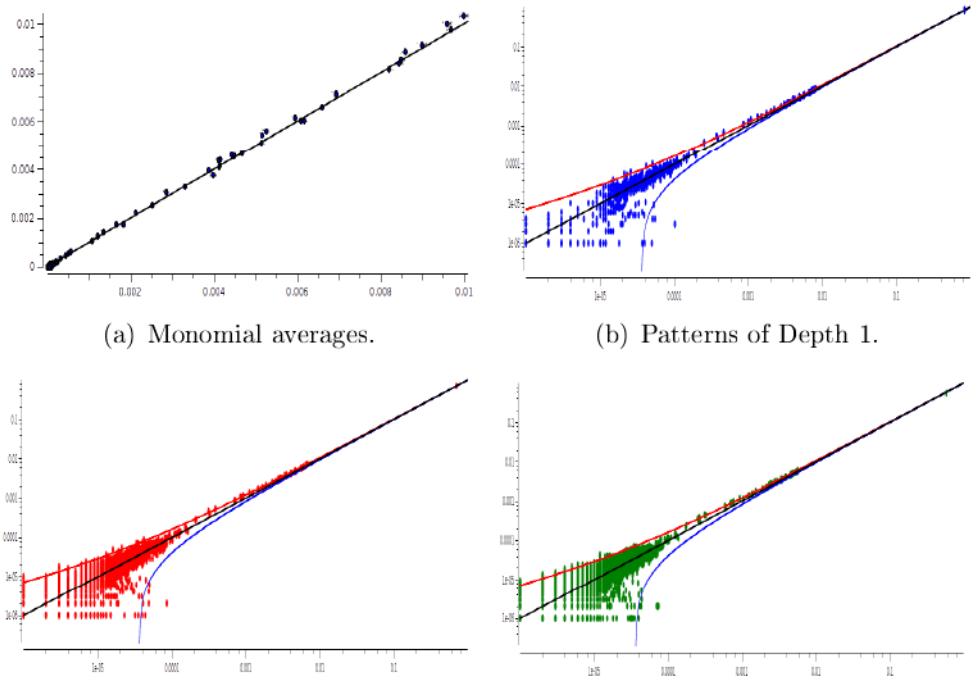
(b)

Figure 3.14: Max distance between the exact value of coefficients and the estimated value, averaged on the set of 10 random potentials for $NR = 60$. (a) Dense spike trains (b) Sparse spike trains.

fluctuations ruled by the central limit theorem. For a block ω_0^D generated by a Gibbs distribution $\mu_{\boldsymbol{\lambda}}$ and having an exact probability $\mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right]$ the empirical probability $\pi_{\omega_s}^{(T)}\left[\omega_0^D\right]$ is a Gaussian random variable with mean $\mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right]$ and mean-square deviation $\sigma = \frac{\sqrt{\mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right]\times\left(1-\mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right]\right)}}{\sqrt{T}}$. The probability that $\pi_{\omega_s}^{(T)}\left[\omega_0^D\right] \in \left[\mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right] - 3\sigma, \mu_{\boldsymbol{\lambda}}\left[\omega_0^D\right] + 3\sigma\right]$ is therefore of about $99,6\%$. This interval is represented by confidence lines spreading around the diagonal.

We have tested the following cases.

1. Spatial case, 40 neurons, ($NR = 40$): Ising model (Eq. 2.23). Figure 3.15 .



(a) Monomial averages.

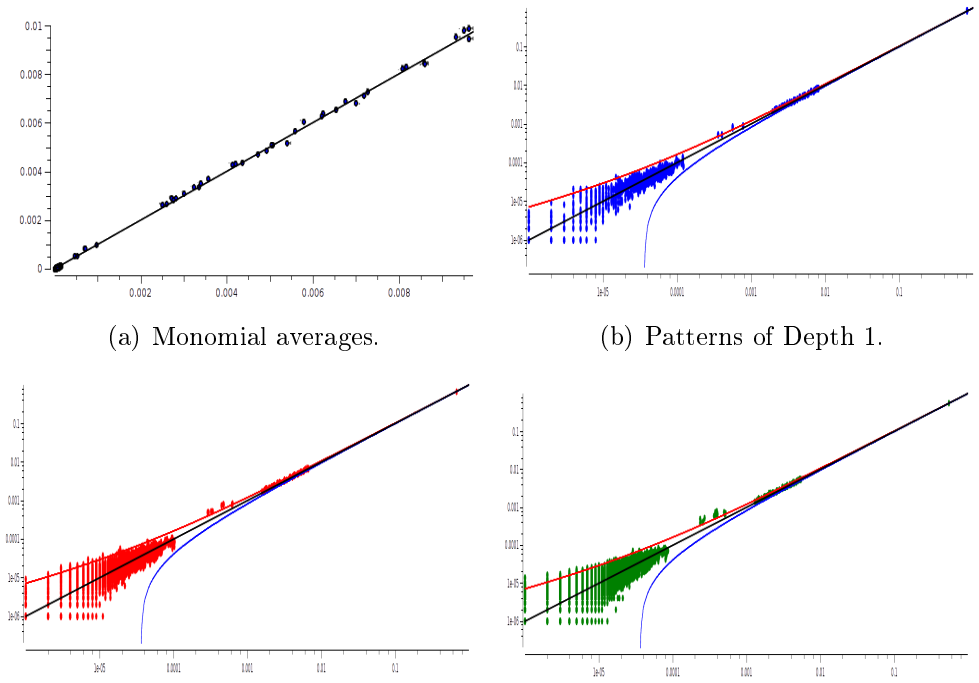
(b) Patterns of Depth 1.

(c) Patterns of Depth 2.

(d) Patterns of Depth 3.

Figure 3.15: Data were generated with an Ising distribution. After fitting with an Ising model, we show the comparison between observed and predicted probabilities of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities of patterns of Depths 1, 2 and 3, respectively. In the four plots, the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The estimated Kullback–Leibler divergence is 0.0107.

2. Spatio-temporal, 40 neurons, $R = 2$ ($NR = 80$): Pairwise model with delays (Eq. 2.42). Figure 3.16



(a) Monomial averages.

(b) Patterns of Depth 1.

(c) Patterns of Depth 2.

(d) Patterns of Depth 3.

Figure 3.16: Data were generated with a pairwise distribution of range $R = 2$. After fitting with a pairwise model of range $R = 2$, we show the comparison between observed and predicted probabilities of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities of patterns of Depths 1, 2 and 3, respectively. In the four plots, the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The estimated Kullback-Leibler divergence is 0.0174.

After fitting the model, the first thing to look at is the monomials averages. We expect that the monomials probabilities predicted by the model are close to the observed ones. Figure 3.16(a) and 3.15(a) show that monomials averages are well predicted the spatial and spatio-temporal case, with 40 neurons. Concerning the prediction of pattern probabilities, Figure 3.15 shows that the fitted model is able to predict well the pattern of depth 1,2 and 3, which is an another argument on the validity of the algorithm.

3.8 Conclusion

This chapter consists on four important contributions of my thesis:

1. A Montecarlo method to compute target Gibbs distributions from a given set of parameters.

2. Parallelization of the Montecarlo method.

3. Extension of the convex-duality principle to compute the parameters for spatio-temporal Maximum Entropy distributions.

4. Computing an approximated value of the Kullback-Leibler divergence for large networks using Montecarlo.

We have seen that computing the target distribution using Montecarlo gives the expected results as shown in Figure 3.7. The parameters fitting algorithm gives excellent results on both small and large scale (Figure 3.10, 3.14, 3.15 and 3.16).

In term of computation time, we are still faced with long computations despite using the parallel version of Montecarlo with clusters. The computation time depends not only of the length of the target distribution, but also the coefficient k, the number of neurons N and the number of parameters L. For instance, on a cluster of 64 cores, we need around 5 minutes to compute a target distribution for 20 neurons with a pairwise r = 2 model, around 10 minutes for 40 neurons with Ising and around 20 minutes for 40 neurons with pairwise R = 2. The time of computing the parameters will be multiplied by the number of parallel and sequential iterations. In general, we run 100-200 parallel iterations and 200-300 sequential to get the convergence. Overall, we need 1 day for a pairwise R=2 on 20 neurons, 3.5 days for an Ising on 40 neurons and 6 days for a pairwise R = 2 on 40 neurons.

As we recommend $k = 30 - 50$ and $N_{times} = 10^6$, the only way yo reduce the computation time is to reduce the number of monomials. In this spot, filtering monomials from the distribution will be twofold:

- Eliminating the risk of infinite fluctuations on the parameters.

- Reduce the computation time.

Chapter 4

EnaS: a new software for large scale spike train analysis

Contents

This chapter presents *EnaS* (Event Neural Assembly Simulation), a software dedicated for analyzing large scale spike trains. *EnaS* is based on the tools presented in chapters 2 and 3, as well as many other statistical modules. *EnaS* exists as a C++ code and a graphical user interface (GUI). The GUI is dedicated to make easy use of the library, especially for End-users, such as biologists. This software is compatible with Linux, Windows and Mac

operating systems and can be used with Matlab. *EnaS* is available online on this web-page `http://enas.gforge.inria.fr/`.

This software project has been initiated in 2007 with a first version, thanks to the works of Prof. Bruno Cessac and Thierry Viéville. As of today, 2014, we are at version 3 with a graphical user interface thanks to the works of several scientists and developers. My main contributions to this software are:

- Implementation and validation of the Montecarlo and Parallel Montecarlo method.
- Implementation of the fitting parameters methods and providing tests.
- Contribution in the design of the software (testing, debugging, etc ...).

EnaS performs both empirical and theoretical statistics. Empirical statistics modules allow the user to manipulate spike trains, load data, select neurons, display the activity and the pattern occurrences and saving plots of the network activity. Theoretical modules allow to choose the hypothesized model, fit it to data and evaluate how empirical and theoretical distribution are close to each other, providing tools to evaluate the models such as Kullback-Leibler divergence and confidence plots. *EnaS* also contains a tool to create empirical distributions from customized theoretical distributions configured by the user.

Thanks to the Montecarlo method [Nasser et al., 2013], EnaS is able to perform statistics for large networks with spatio-temporal constraints. EnaS can be run in parallel processing. We have implemented the possibility of parallel computing using OpenMP and MPI. Furthermore, the user can choose between computing methods: transfer matrix (for small size network, $NR < 20$) or Montecarlo (for larger networks). The user is also given the freedom to choose the potential type (Bernoulli, Ising, Pairwise, triplets as well as completely customized potentials using a monomial file [1]). These points are developed below in details.

4.1 *EnaS*: concept and design

EnaS implements functionalities to manipulate data, perform statistics and display results of empirical and theoretical computations. Figure 4.1 shows the three major cores of *EnaS* :

[1] The monomial file contains the set of monomials that constitute the features of the Gibbs potential

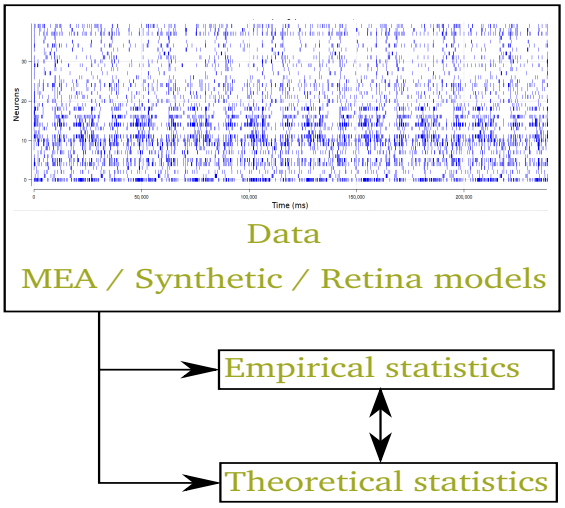


Figure 4.1: The concept of EnaS. *EnaS* reads data in 3 formats (details in Figure 4.3) and perform empirical and theoretical analysis. The empirical analysis are performed directly on data, such as counting patterns occurrences and computing firing rates are directly measured on the data (for example, see Figure 2.2). Theoretical analysis requires defining the models and the fitting parameters as well as the empirical measures.

- Performing empirical statistics on spike trains.
- Fitting of general spatio-temporal statistical models.
- Display results of statistics.

The *EnaS* library contains about 40 classes coded with C++. Some classes are dedicated to manage spike train data and empirical measurement Others classes are dedicated to define the models and fit the parameters. We also introduced some classes dedicated to purely technical tasks, like those using graphics (Gnuplot for instance) or other standard libraries for printing and generating random numbers. We adopted the C++ language for many practical reasons:

1. Speed: Because of its intermediate programming level, C++ ensures a speed in computation and a compromise between speed and coding easiness.
2. Inheritance [2]: the concept by which derived classes (or sub-classes) can inherit attributes and behaviors from super (or parent) classes. This

[2]Inheritance is the process by which new classes called derived classes are created from

concept implies a hierarchy in the code and allows its reusability (from parent to sub classes). Hence, this allows saving time and effort, keeping the library more structured and prevents from adding redundant codes at many places in the library.

3. Popularity: C++ is one of the most popular languages and is implemented on a wide variety of hardware and operating system platforms. Installing *EnaS* requires installing free libraries which are available online but does not need any IDE [3] such as Matlab, Visual studio, etc ...

4. Possibility of interfacing the library with other programming languages (i.e., python and java) as well as IDEs.

Note on classes and objects The Object Oriented Programming (OPP) offers the possibility to create objects and classes. This has several advantages for the developers. For instance: the reusability, the inheritance and the organization of the code. In *EnaS*, for example, we can consider that as spike train is an object which has attributes and functions.

Attributes for such objects are:

- Number of neurons.
- Length of the spike train.
- Binning value.
- ...

Functions for such objects could be:

- Computing the firing rate.
- Concatenating two or more spike trains.
- Extracting subsets of neurons.
- ...

Another example of objects is the model, which can have the list of monomials as an attribute and the method of fitting parameters as a function.

existing classes called "base classes". The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.

[3] IDE: Integrated Development Environment.

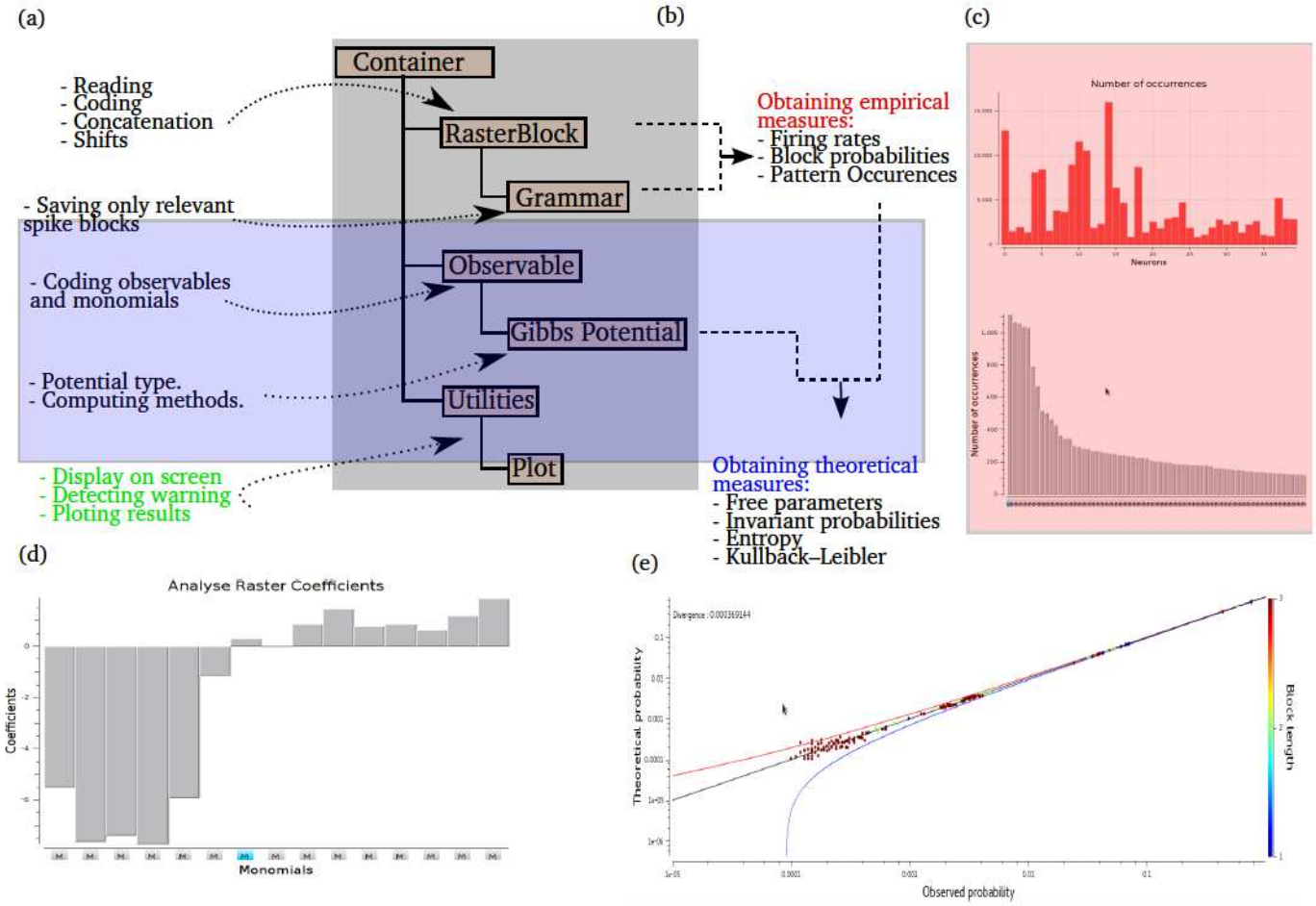


Figure 4.2: *EnaS* functionalities. (a) The main tasks of each (sub) module. (b) The main 3 functionalities: *Rasterblock*, *Observable* and *Utilities*. They belong to the parent library class *container*. Each functionality contains several sub-classes (derived classes). Here we present only the main sub-classes. For instance, the *RasterBlock* is responsible for coding the raster data, concatenating and making algebraic shifts (the iterator facility, Figure 4.4). The *Grammar* sub-module that derives (and inherits) from *RasterBlock* is responsible for refining the *RasterBlock* structure by identifying only the spike patterns that appear in the data and determine the transition probabilities. (c) Examples on some empirical statistics using *Rasterblock* (Firing rates (top) and patterns occurrences (bottom)). (d) Coefficients display and (e) confidence plots: results of the theoretical statistics performed by the *Observable/Gibbs potential* functionality.

4.2 The main functionalities in *EnaS*

In the following, we will explain in details the main functionalities of *EnaS*:

- RasterBlock (and its derived class Grammar).
- Observable (and its derived class Gibbs potential).
- Utilities (and its derived class Plot).

Figure 4.2 shows the functionalities tree (where we emphasize the inheritance property), their main roles and some of the results we can obtain by using them.

4.2.1 RasterBlock

RasterBlock is the *EnaS* input data manager. Data are supposed to be event-based. Events happen when a neuron fires a spike, which leads to the event pair $(neuron, time)$. Hence, event-based data only contains firing event pairs. The supported data (Figure 4.3) format are:

- Unit-time (two columns data, where the first one corresponds to the firing cell index and the second one corresponds to the time stamps).
- Time-unit (like unit-time, but the first column corresponds to the time stamp and the second one corresponds to the cell index).
- Unit-by-line (each line contains the time stamps of one neuron); clearly, data are event-based, e.g., only time stamps when the neuron fire exists.
- Neuron-per-column with MEA coordinates, where each column contains the events of 1 neurons with its MEA coordinates (Figure 4.13).

RasterBlock coding strategy: We used C++ sets in order to store the spikes data. Sets are containers that store elements (events) following a specific order. We used sets because we are interested in a structure that allows fast access to the stored elements. Sets are the fastest containers that C++ provide. The speedy access to the elements in the sets is of high importance because usually spike train acquisitions are heavy and when it comes to counting the specific events and search for them in the set, one needs to traverse a lot of memory cells.

EnaS implements a specific iterator. Like iteration over a real integer number, we need to iterate over a variable of type RasterBlock. Figure 4.4 gives a typical example of iteration over a RasterBlock of size 3 neurons

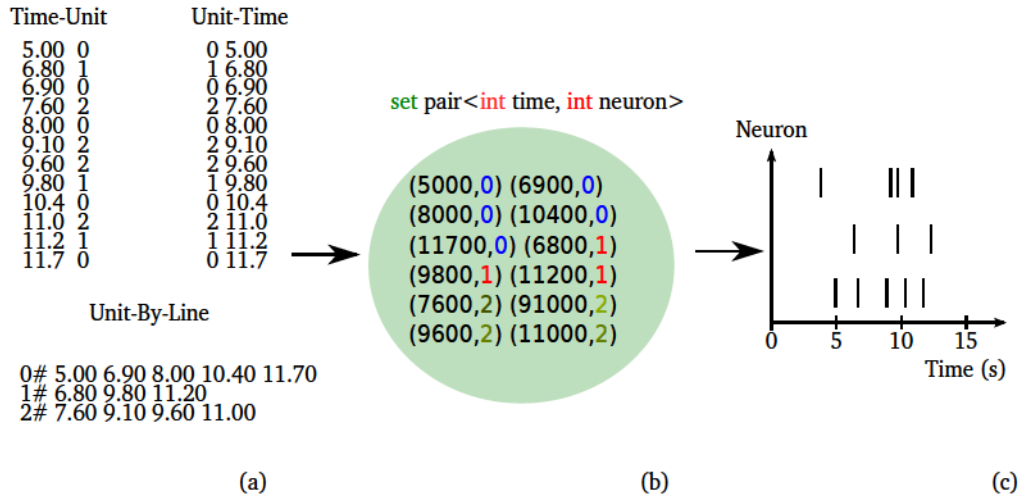


Figure 4.3: Data management in EnaS. (a) *EnaS* can read spike trains with one of the following three formats: time-unit, unit-time and unit-by-line. We defined those formats as acceptable formats because they are the most used in the community. When importing the spike train file, the user should precise what is the format of the data. (b) *EnaS* stores those data in C++ sets and displays them. The ***C++ set*** features sorts automatically events in increasing order with respect to neuron index and time stamp. The time stamps are processed with respect to the sampling period and finally transformed to integer values for technical reasons (at many stages in processing the spike trains, we might need to compare time stamps of two neurons). The comparison of double values is not precise because of the floating points. Hence, we transform them to integer values). (c) The spike train drawn using plotting facilities provided by *EnaS*.

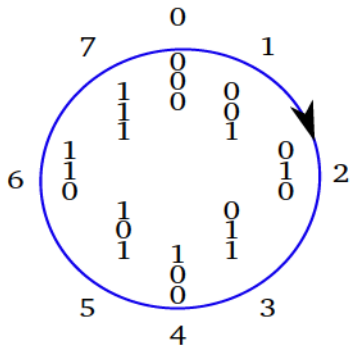


Figure 4.4: The **Iterator** facility provided by RasterBlock. It is a tool that implements an iterator over a RasterBlock. The integer numbers outside the circle are only set for clarification. Like iteration over integer number($0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$), the iterator perform an incrementation of the RasterBlock which is a Boolean array ($000 \rightarrow 001 \rightarrow 010 \rightarrow 100 \rightarrow 101 \rightarrow 110 \rightarrow 111$). It begin with 000 and at each iteration. It performs simple incrementation on the correspondent Boolean array.

and 1 time-step. The iterator follows the order of Boolean representation of integers. We use this iterator in EnaS to scan spiking patterns in order to compute their probabilities.

RasterBlock functions Apart from reading data, the RasterBlock functionality offers many other tools:

- Extracting sub-spike trains, those who correspond to neurons the user selects, i.e., creating another spike train with less number of neurons specified by the user.

- Extracting the activity between two times from the spike train (t_1 = "beginning time" and t_2 = "ending time" are precised by the user) . The user can analyze or visualize this specific part of the spike train if desired.

- Concatenating two or more raster blocks together. An example of using this facility is when we have a spike train with many stimuli and want to make spike trains that corresponds each to one of the stimulus.

- Binning: the sampling period for a spike train. Basically, spike trains are acquired at various sampling period (e.g. 0.1 or 1 ms). *EnaS* allows

the user to change this sampling period if the user desires. Figure 4.5 gives two example of the binning process.

- Creating random spike trains with a given Gibbs distribution. The idea consists on generating a spike train from a set of monomials and parameters. This is nothing but generating a target distribution with Montecarlo (this issue is detailed in Figure 3.1). It is also possible to generate the target distribution with the transfer matrix, but only for small networks or small number of neurons.

- Comparing two raster blocks by comparing the mutual events of both raster blocks.

- Computing the Hamming distance between two data sets: measuring how dissimilar they are.

$$D(\omega, \omega') = \sum_{t=0}^{t=T-1} \sum_{i=0}^{i=N-1} |\omega_i(t) - \omega'_i(t)| \tag{4.1}$$

- Read and/or change a neuron state at any time: tells if the neuron fire or not and the possibility of toggling the value of an event.

- Performing periodic and non periodic shifts on the spike blocks. The spike block ω_0^{R-1} of size R is a set of consecutive R spike patterns ($\omega(0)$, $\omega(1)$ $\ldots\omega(R)$). The shift consists on moving all the spike patterns toward the left. $\omega(1)$ replaces $\omega(0)$, $\omega(2)$ replaces $\omega(1)$ and so on. The difference between periodic and no periodic shit is the following: in the periodic case, when $\omega(0)$ gets shifted, it replaces $\omega(R)$, however, in the non periodic case $\omega(R)$ is replaced by $\omega(R+1)$ and $\omega(0)$ goes away. This functionality has been used to infer the form of the Gibbs potential from a spike train (e.g., [Cofré and Cessac, 2013]).

4.2.2 Grammar

The Grammar sub-functionality is derived from RasterBlock. It is an analogy with language grammar, which represents the speaking or writing rules. Likewise, for a spike train, which would contain 2^{NR} possible transitions, the grammar is a tool to conclude, from a spike train, the rule of observed legal transition (see section 2.2.4). In other words, the grammar scans all the spike train and maps only the observed transition blocks. It represents the obtained value in a C++ map.

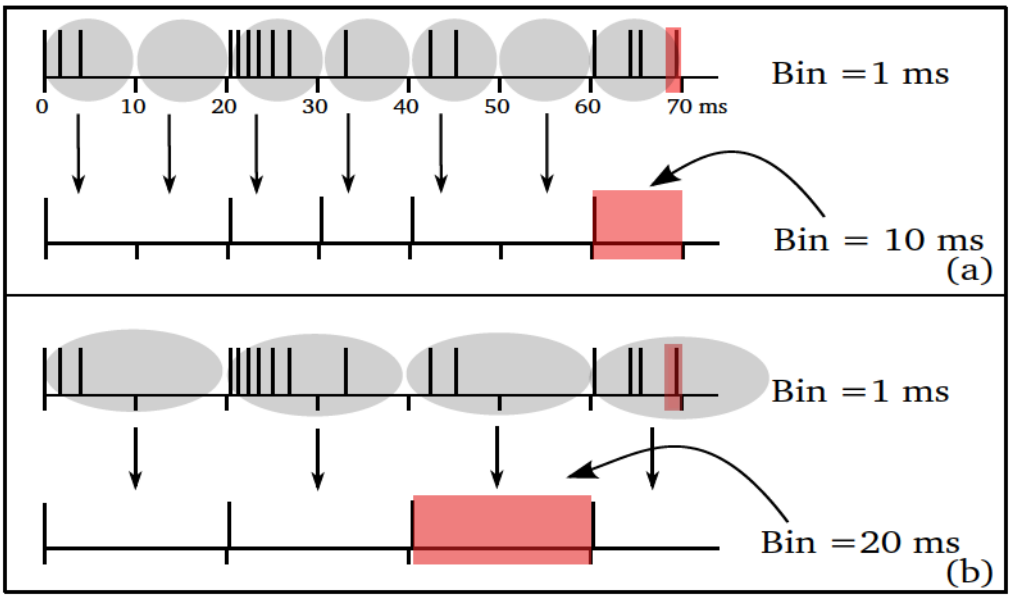


Figure 4.5: The binning concept. Binning is the process by which we change the basic sampling period of a raster. Basic sampling period is the one used during the acquisitions. Usually, it is in the order of 0.1 or 1 ms. (a) binning at 10 ms. The process consists on scanning time windows of 10 ms along the raster and consider a spike whenever the time window contains at least one spike. If the time window does not contain any spike, then the binned raster event will be set to 0. (b) binning at 20 ms. Same like (a), but we scan time windows of 20 ms.

The transition block represents the states of the whole network at two adjacent time window: the past (of size D) called prefix and the current pattern that represent the state of the network (of size 1), called suffix. Prefix and suffix together make a RasterBlock of size D+1 = R. Hence, we look at transitions by looking directly at blocks occurrences of range R. For example, in a network where $D = 3$, we look at the occurrences of blocks of size 3. Hence, the appearance of the block $\left(\begin{smallmatrix}0&1&1\\1&0&0\\0&0&1\end{smallmatrix}\right)$ means that there is a transition from $\left(\begin{smallmatrix}0&1\\1&0\\0&0\end{smallmatrix}\right)$ to $\left(\begin{smallmatrix}1\\0\\1\end{smallmatrix}\right)$.

It is important to know how the Grammar stores the blocks in order to understand the efficiency of the structure as well as its importance. As for sets, the grammar structure contains key values and mapped values. A key value corresponds to a prefix of size D. The mapped value corresponds to a set of suffixes of size 1. This set of suffixes corresponds to the legal transition that are found in the raster block. This gives a tree which is clearly illustrated in Figure 4.6.

For N neurons and R time steps there are 2^{NR} possible transitions whereas in an experimental raster of length T there are $T - R$ transitions. Typically, $T = 10^{6-7}$ whereas $NR = 100$. For example, in a network of 100 neurons, withing a range of 3 time steps, we have 2^{300} possible transitions. However, in real acquisitions, the detected transactions constitute a small percentage among all the possible ones. Take for example the short spike train presented in Figure 4.2.1 with 3 neurons. Let us display the zeros and ones of this spike train. It reads:

$Neuron\#0$: 0001000001100
$Neuron\#1$: 0000001001001
$Neuron\#2$: 0000101010110

Each line corresponds to the activity of a single neuron and each column corresponds to a 1 time step. Theoretically, this spike train would contain 512 possible transitions, which are the following:

	000	000	000	000		111
Transition blocks	000	000	100	100	...	111
	000	100	000	100		111
Integer equivalent	0	1	2	3	...	512

However, in this raster block, we find **only 8 possible transitions blocks**, which are:

000	001	010	100	000	000	000	001	011	110	100
000	000	000	000	001	010	100	001	010	100	001
000	000	001	010	101	010	101	010	101	011	110

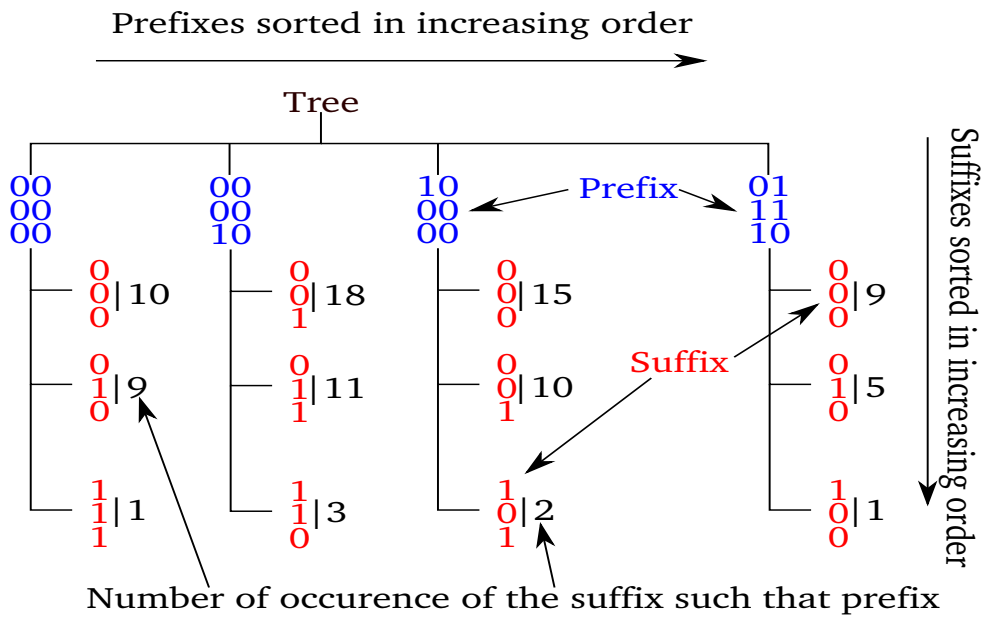


Figure 4.6: C++ set for spike train grammar. The Grammar facility generates a C++ set that corresponds to transitions in the spike train. A transition block is a concatenation between a prefix and a suffix. The suffix is always a spike pattern of size 1 time step. The prefix size depends on the memory of the potential, $D = R - 1$. The Grammar generates a tree where keys correspond to the prefixes. For each key / prefix, we associate a set of mapped values which are the suffixes. This will allow, not only to find the transition easily, but also to compute their occurrence. For instance, in order to count the occurrence of the transition block $\left(\begin{smallmatrix}0&0&0\\0&0&0\\0&0&0\end{smallmatrix}\right)$, the Grammar finds the prefix $\left(\begin{smallmatrix}0&0\\0&0\\0&0\end{smallmatrix}\right)$ and then count how many times their $\left(\begin{smallmatrix}0\\0\\0\end{smallmatrix}\right)$ happens. Doing this instead of scanning again the spike train allows us to gain efficiency in the computation and avoiding long computation periods.

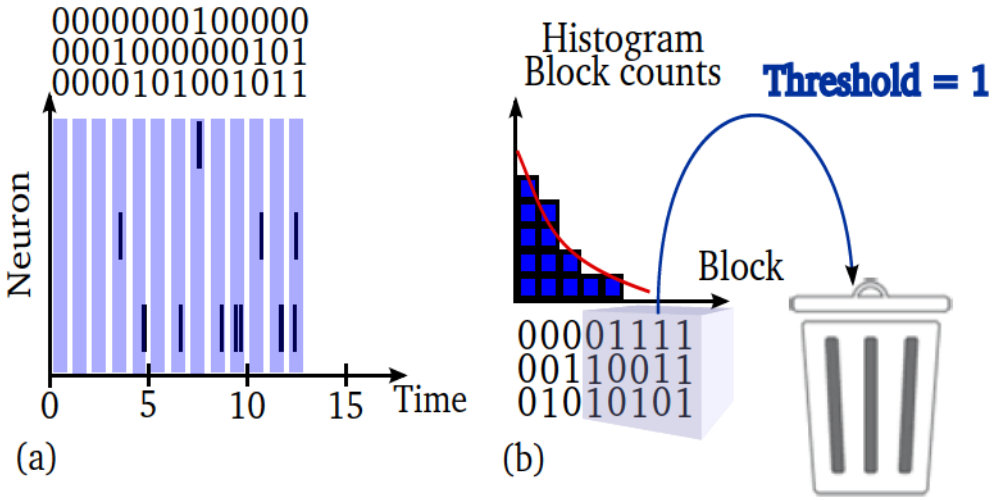


Figure 4.7: An example of the Grammar facility. (a) An example of a 13 time-steps spike train for 3 neurons. (b) The Grammar counts the occurrences of all possible blocks (as shown in the histogram/lookup table for illustration), and then saves only the patterns whose occurrences is bigger than the threshold (threshold = 1 in this example). The threshold allows to cut off the patterns considered by the user as statistically irrelevant.

Note that, in the 2 tables above, each transition block corresponds to a concatenation between a prefix of size $D = R - 1 = 2$ and a suffix of size 1.

Therefore, the Grammar figures out the possible/legal transition that happen in a spike train and creates a look-up table of transitions-probabilities using C++ maps. This reduces the space given to store the possible transactions and their occurrences. In addition, it constitutes a structure on which we rely to compute other empirical averages, such as conditional probabilities. The key values in the map are the transition blocks and the mapped values are their probability. The structure obtained from such sorting will look like the tree in the figure 4.6.

Another main issue about Grammar is thresholding. After counting the possible transitions, one can eliminate the transitions with small occurrences because they are not statistically relevant. Figure 4.7 explains the concept.

To summarize, the main task of the Grammar functionality is to help creating an organized structure of spike blocks in order to compute the empirical (joint, conditional and marginal) probabilities of patterns and features.

4.2.3 Observables & Gibbs potential

This functionality offers tools to design a statistical model: monomials and parameters. Gibbs Potential inherits some attributes from the observables class. Figure 4.8 show how Gibbs potential and Observables are connected together. The main tools that Gibbs Potential and Observable offer are:

- Managing observables: definition and printing the observable in an explicit way ($((\omega_1(0)-1)\omega_2(0)\omega_1(2)$ for instance).
- Defining a Gibbs potential:
  - Parametric : A set of observables that could be chosen and edited by the user ($\mathcal{H} = \sum_l \lambda_l m_l$).
  - Spatial-based potential: *EnaS* offers the possibility of configuring potentials with pre-implemented shapes, such as:
    * Bernoulli - where neurons are considered to fire independently: Eq. 2.22.
    * Ising - Only firing rates and instantaneous pairwise interaction are considered: Eq. 2.23
    * Triplet - triplets instantaneous interactions are considered: Eq. 2.41
  - Spatio-temporal based potential, such as:
    * Pairwise with delays - It contains instantaneous as well as temporal interactions: Eq. 2.42.
    * Triplets with delays - triplets interactions with delays and so on.
- Computing the theoretical conditional probabilities.
- Computing the entropy and the Kullback-Leibler divergence.
- Generating a spike train from given a probability distribution.
- Computing a target probability distribution with the Montecarlo technique.
- Fitting the parameters λ_l.

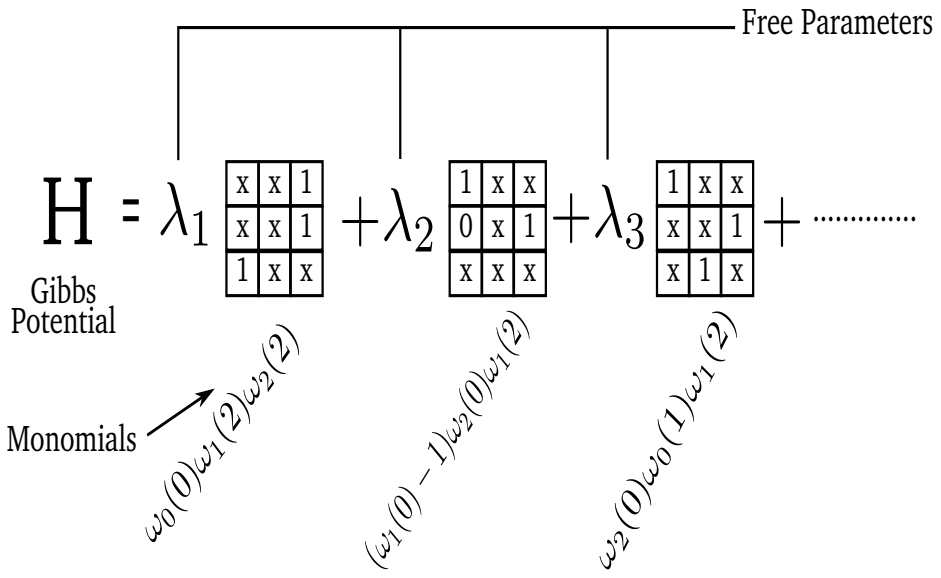


Figure 4.8: Gibbs Potential and Observables. Each Gibbs potential is defined by the monomials and their correspondent free parameters λ_ls. We presented monomials as masks in the equation in order to clarify how those monomials are implied in shaping the Gibbs Potential. The free parameter associated to a monomial represents the statistical weight of such monomial. For example, the parameters related to firing rates are negative and the more negative they are the less active the cells is.

4.3 Codes development awareness

We implemented the functionalities with precise coding and naming rules, ensuring the robustness and usability of the code. We also optimized the codes in such a way it takes the minimum possible time when one runs a task using *EnaS*. Since *EnaS* has been developed by many scientists, we determined coding and naming rules that each co-developer should follow. We also created common network (subversion repository) where co-developers can add and change the library codes.

4.3.1 Naming rules

The naming rules determine the conventions on how to choose, write and administer names for all entities over which the programmer has control. This guarantees that programs are easier to understand, read and maintain. The rules are:

- The name of the header file should be the same as the name of the class it defines, with a suffix ".h" appended. Example : The header file for the class **RasterBlock** would have the name **RasterBlock.h**.

- The name of the implementation file should be the same as the name of the class it implements, with a project dependent suffix appended. Example : The implementation file for the class **RasterBlock** would have the name **RasterBlock.cpp**. The template implementation file (if exists) for the class **RasterBlock** would have the name **RasterBlock.tpp**.

- Co-developer should use pronounceable names, or acronyms used in the experiments.

- Using names that are English and self-descriptive.

- Names of classes, methods and important variables should be chosen with care, and should be meaningful. Abbreviations are to be avoided, except where they are widely accepted.

- Do not use identifiers that begin with an underscore.

4.3.2 Coding rules

We determined a number of coding rule to manage and control the content of the code. Organization of the code, control flow [4], object life cycle [5], conversions, object-oriented programming, error handling, parts of C++ to avoid, portability, are all examples of issues that are covered withing the coding rules. The purpose of the following rules is to highlight some useful ways to exploit the features of the programming language, and to identify some common or potential errors to avoid:

- Each header file should be self-contained: If a header file is self-contained, nothing more than the inclusion of the single header file is needed to use the full interface of the class defined.
- Avoid unnecessary inclusion: This is necessary to guarantee that the dependencies present in the implementations are only those foreseen in the design. Example : unnecessary inclusion in the header file.

```
file A.h:
#include "B.h"

file C.h:
#include "B.h" // NOT necessary, avoid
#include "A.h"
```

- Header files should begin and end with multiple-inclusion protection.

```
#ifndef enas_ClassName_h
#define enas_ClassName_h
// The text of the header goes in here ...
#endif // enas_ClassName_h
```

- Use forward declaration instead of including a header file, if this is sufficient

```
class Line
class Point {
public :
// method definition
}
Here it is sufficient to say that Line is a class, without
giving details which are inside its header. This saves
time in compilation and avoids an apparent dependency
upon the Line header file.
```

[4] The C/C++ control statements: If, elseif, switch, break, while, for

[5] The object life cycle states the rules of programming an object by assigning and instructor, destructor, and by managing the copying of the objects

- Each header file should contain one class declaration only. This makes easier to read your source code files. This also improves the version control of the files for example the file containing a stable class declaration can be committed and not changed anymore

- Implementation files should hold the member function definition for a single class as defined in the corresponding header file. This is for the same reason as above.

- Each header file should never contain ″using namespace ″directive.

- Each template implementation should be called at the end of the corresponding header file.

```
ClassName.h :
#ifndef enas_ClassName_h
#define enas_ClassName_h
// The text of the header goes in here ...
#include "ClassName.tpp"
#endif // enas_ClassName_h
ClassName.tpp :
#ifndef enas_ClassName_tpp
#define enas_ClassName_tpp
// The text of the template implementation goes in
here ...
#endif // enas_ClassName_tpp
```

- Enas uses one namespace that should be define in each Header file (.h) and Implementation file (.cpp) of the library. The namespace should appear after each include file 3 and before the class declaration. To export from a DLL (windows compilation) all of the public data members and member functions in a class, the keyword ENAS_EXPORT must appear to the left of the class name. Here is an example with ″RasterBlock.h ″:

```
#ifndef enas_RasterBlock_h
#define enas_RasterBlock_h
#include "sys.h"
.
.
#ifdef NAMESPACE
namespace enas {
#endif
class ENAS_EXPORT RasterBlock {
// The text of the class goes in here ...
}
#ifdef NAMESPACE
```

```
    }
    #endif
    #endif // enas_RasterBlock_h
```

- Comment each header file using **Doxygen** documentation syntax.

4.4 *EnaS*: The graphical user interface

The implemented tools in C++ could be used by any programmer who desires to analyze spike trains. In addition to this library, we implemented a graphical user interface (GUI) where the user can exploit all the resources in *EnaS* without writing C++ code. The three main reasons for creating the GUI are:

1. Make the analysis easier. With the GUI, we only need to click on buttons and edit boxes in order to analyze the data. On the opposite side, writing the C++ codes are time-consuming and implies debugging all the time.

2. Creating a tool for non programmers. Biologists and experimentalists who desire to analyze the spiking data, have a tool that makes them analyze the data without any knowledge in programming. We did also an effort in order to make the GUI notations understood by scientist.

3. Interactive tool and display: the GUI allows the user to interact with the data and the models using the peripherals of the computer. We can load data, run computations, display data and save the results.

The functionalities of GUI are:

- Spike train management window (Figure 4.9).
- Pattern Histogram and filtering window (Figure 4.10).
- Probabilistic modeling window (Figure 4.11).
- The spike/stimulus display window (Figure 4.12).

We designed *EnaS* in such a way that empirical statistics are separated from modeling so that the user is able to understand easily the use of the software. In addition, we added several interactive tools that allow the user to save and load data. The user can also save the whole project in a file of extension *.ens*. The idea of creating a Graphical User Interface came from

our interest in creating a common framework for all the neuroscientists to analyze spike train. A very ambitious project, where methods now consists of displaying empirical data and fitting MaxEnt models in a highly interactive environment. Section 6.4, "The future of *EnaS*" in the last chapter is dedicated to discuss a future opportunity with this software.

4.4.1 Spike train management window

This window is responsible of managing the spike train, whether they are saved on computer or generated with *EnaS*. The tasks that this window (Figure 4.9) can perform are:

- Reading spike trains with a specific sampling period and specific format.
- Displaying spike trains with a specific binning value.
- Sorting neurons by index or firing rates (and possibility of saving the selected neurons data in a separate file).
- Possibility of analyzing selected number of neurons (and not only the whole raster).
- Generation of artificial raster with customized probability distribution.

4.4.2 Pattern Histogram and filtering window

This module is dedicated to display the occurrence of patterns and spike blocks. It also allow us to filter pattern that appears less than a certain number of times that is identified by the user. More precisely, this window allows us to:

- Display the number of occurrences of all the possible spike block until a certain time size that is precised by the user.
- Display the occurrences of features. In this case, the user has to choose the model. For instance, if the user chooses Bernoulli, then *EnaS* will display only firing rates. Likely, if the user chooses Ising, *EnaS* displays the rates and the instantaneous pairwise interactions.
- Filter the monomials. If the user to take into account the features that appears only more than a certain number of times in the raster, he has to precise the threshold and *EnaS* will filter the monomials. The user can use those non-filtered monomials as features in order to construct a

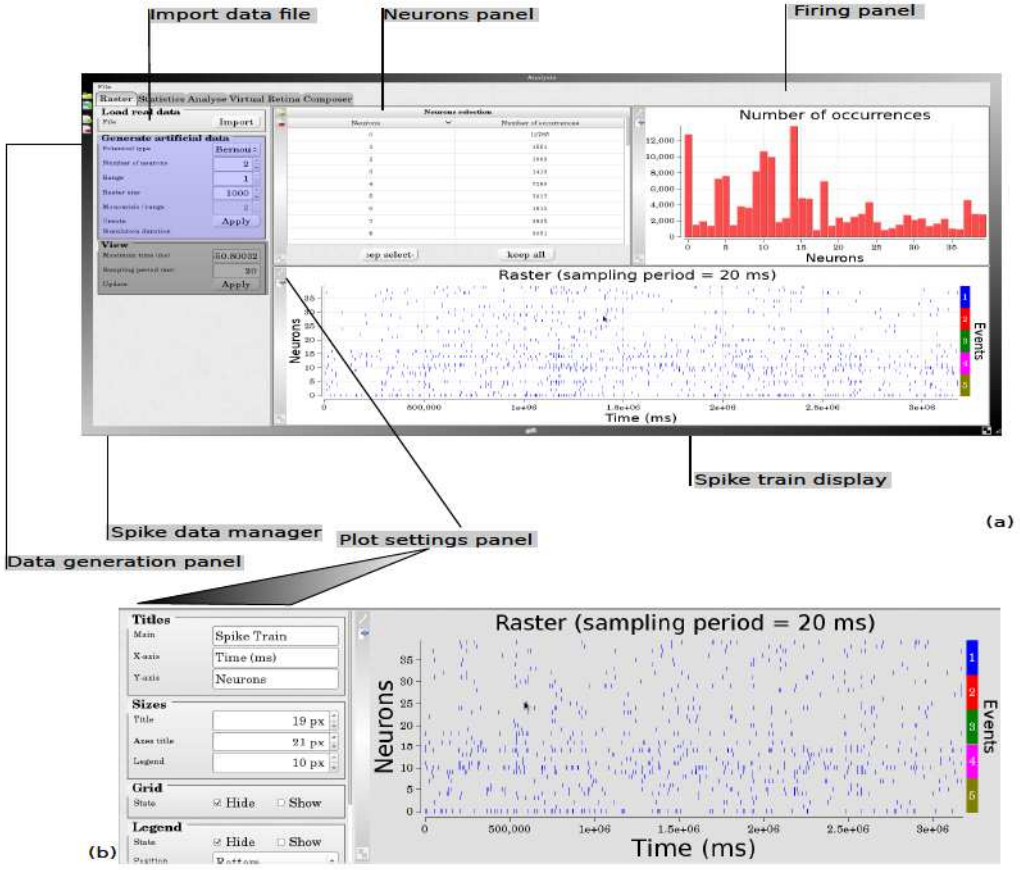


Figure 4.9: The empirical statistics module. This module allows to manage spike train data. It contains the **import panel** that allows to load a spike train. The user can also precise the format and sampling time of the data. The **neuron panel** displays the neurons and their occurrences. It also allows sorting neurons by activity, selecting some neurons and saving data of the selected neurons with the format of choice. The **firing panel** displays the histogram of neurons activities. The **spike data manager** allows to change the binning value and change the maximum time of the displayed raster. The chosen neurons are displayed as a spike train in the right-down part of the window. Spikes are displayed in several colors (See the event scale at right of the spike train) because of binning. Indeed, the colors corresponds to the number of spike found in one time bin. Finally, the data generation panel allows to create spike train with specific probability distributions. The user sets the distribution shape (Bernoulli, pairwise, ...), network size, range and raster length. (b) The plot setting panel is dedicated to control the outlook of the plot. The user can change the font size, axis name and colors. In this example, the spike train shown in (a) is the same of the one shown in (b) but with different outlook settings.

model that could be fitted in the probabilistic modeling window (Figure 4.11

4.4.3 Probabilistic modeling module

The probabilistic modeling module (Figure 4.11) allows the user to choose a hypothesized model and fit it to the data configured in the Empirical statistics modules (Figure 4.9). The potential configuration panel is dedicated to choose the configuration of the hypothesized model (Shape, range, length of Montecarlo chain). The computing configuration panel allows the user to choose the computing technique (Montecarlo or transfer matrix). If the computing technique is Montecarlo, the user has also the possibility to tune other variables related for parameters fitting. For instance, the convergence bound and threshold for updating the Montecarlo distribution. The user has also the possibility to customize his own potential by choosing a monomial file via the monomial panel.

In this window, *EnaS* displays the comparison of patterns probabilities in data with those predicted by the model. This graph contains the confidence bounds (the area where the error is permitted), the equality axis and the divergence value. The last 3 tools helps the user to judge whether the models fits to the data or not. The monomial histogram displays the values of the free parameters that corresponds to the potential monomial. If we click on a monomial in this histogram, we will see right to the histogram the binary form that corresponds to such monomial. Finally, the form of the Gibbs potential with the value of free parameters and the analytic expressions of monomials appears on the top-right of the window. The error, displayed in red, is the Hellinger distance [6] is computed at each iteration in parameters fitting in the case of Montecarlo.

4.4.4 The spike/stimulus display window

During the development cycle of *EnaS*, we asked the following question: Is it enough to fit the models without any information about the cell nature and position? The answer is obviously, no. For that, we wanted to add another functionality in *EnaS* that allows to show the stimulus and instantaneously the activity of cells respecting their repetitive fields. The ultimate goal from this module is to be able to characterize cell type and in order to select subset of the whole network on which we test models (not exclusively MaxEnt models, but also other. See section 4.5 for more details).

[6] For details about the Hellinger Distance, see section 3.6.3

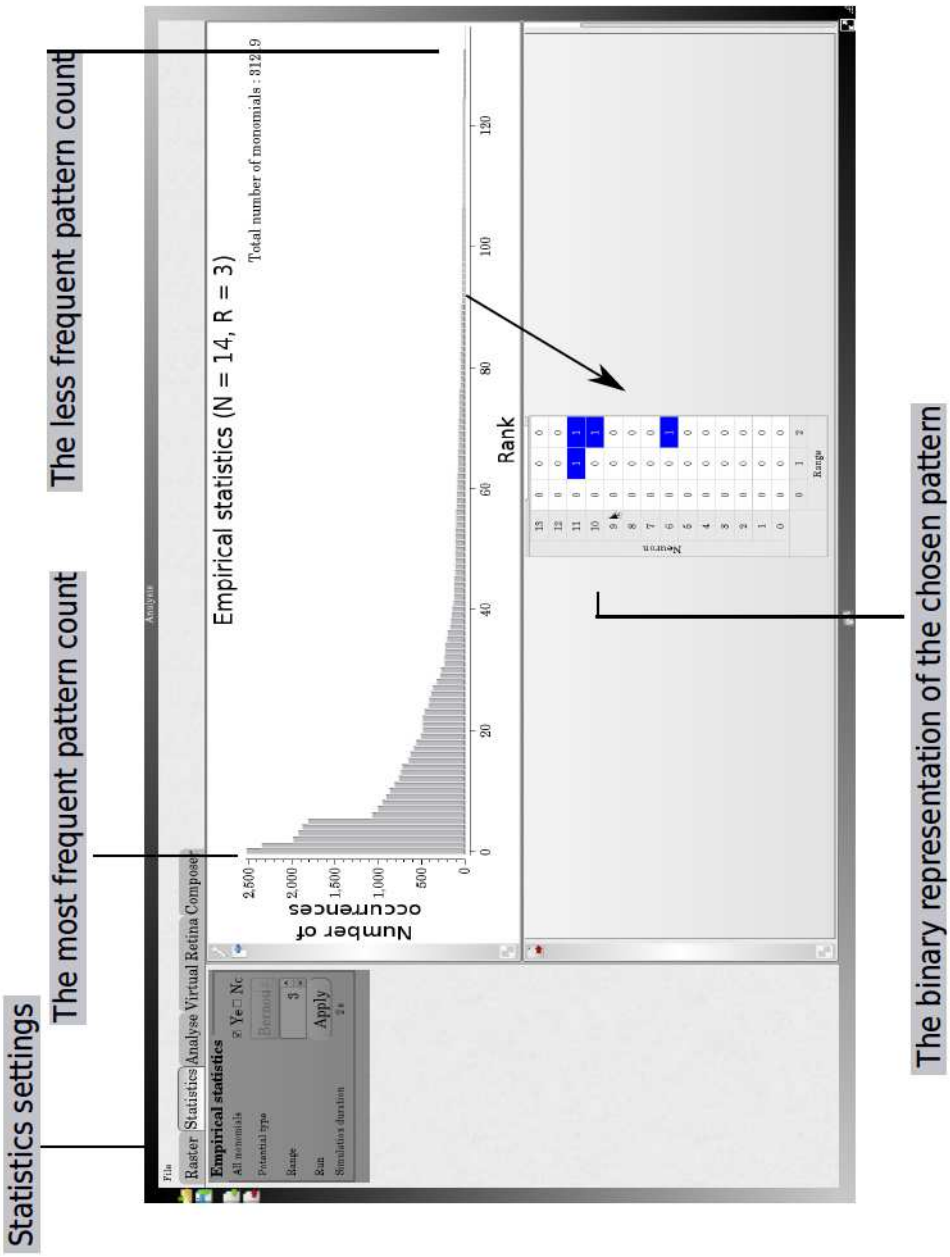


Figure 4.10: The Histogram module. The user chooses the maximum range of patterns (e.g. the maximum size of time-steps of patterns) wanted in the histogram. *EnaS* sorts the occurrence probabilities and draw them in a histogram from the highest value until the lowest one. The x-axis represents the measured pattern. The user can click on the letter "M" in order to see the binary representation of the pattern. In this example, the chosen pattern corresponds to: the neurons #6, #10 and # fire simultaneously right after the neuron #11.

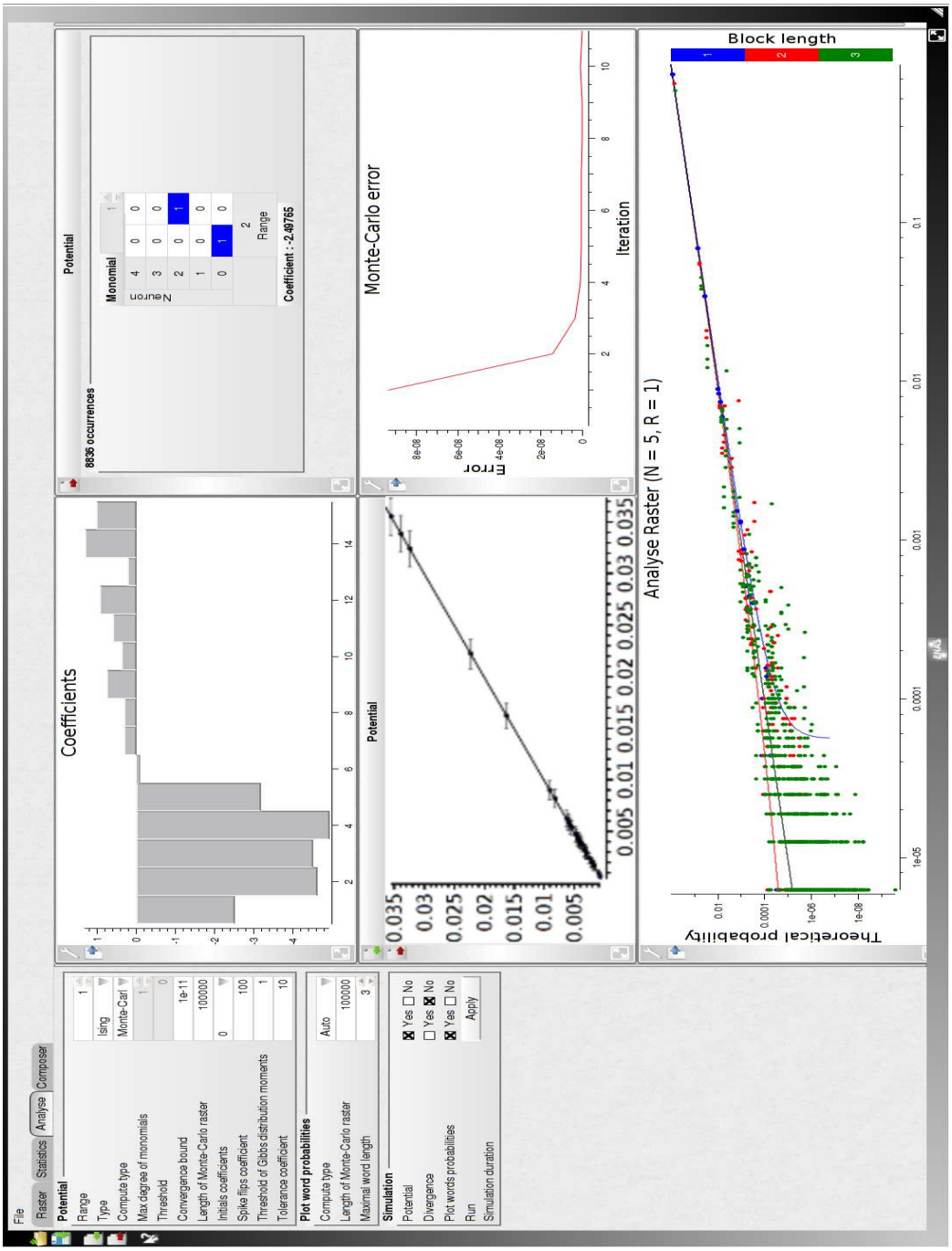


Figure 4.11: The probabilistic modeling module. This the window of parameters fitting. The user configures the hypothesized model (at left of the window) and sees the results of parameters fitting at the right side. *EnaS* display the results in a highly interactive way: The error (red and blue curves that corresponds to the confidence bounds) is displayed at each iteration and the user can stop the computation if he sees a dramatic increase in error. Monomials and their averages are displayed also, where the use can choose the monomial to display. Finally, pattern probabilities, those predicted by the model and those measured on the spike train are equally displayed.

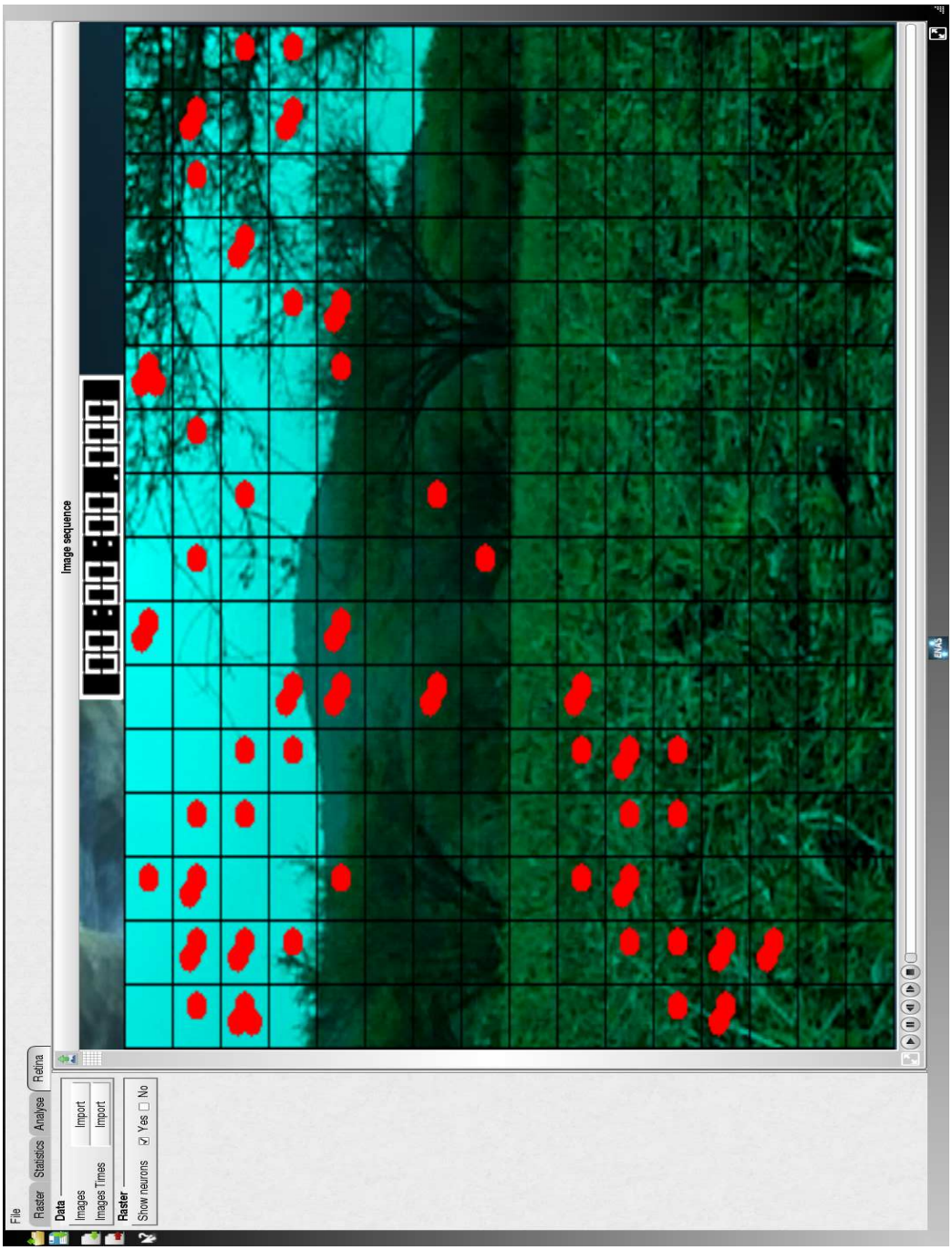


Figure 4.12: The retina module. The activity of neurons is shown in real time in synchronization with the stimulus on the retina. Data courtesy: Adrian Palacios and Maria-Jose Escobar (Centro de Neurosciencia de Valparaiso).

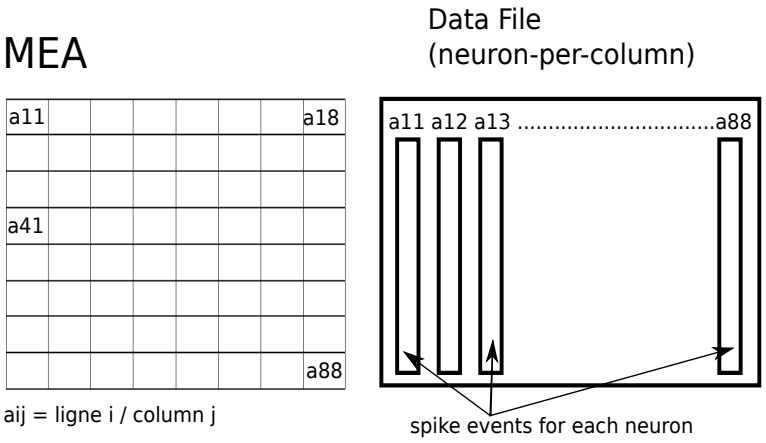


Figure 4.13: Illustration of data file with MEA coordinates (neuron-per-column). Each column contains the name of the neurons (which contains explicitly its MEA coordinates) and its corresponding events.

For now, the module is dedicated to display the response vs stimulus. The response has two attributes: the activity of cells and their receptive field. In the current version, we consider that the receptive fields are the same as the neurons coordinates in the MEA. But, we are working on implementing a functionality to compute the receptive fields of cell and will introduce it in the next version of *EnaS*. The synchronization between the stimulus and network activity display is assured by a specific file that contains the synchronization information.

As a consequence, the user has to have the following files in order to run the animation:

- A file that contains the spike train along with the cells position in the MEA (Figure 4.13.)

- The directory of the frames: the frames should be named in an increasing order. For example, if we have 300 frames, the images should be named IMG001.jpg, IMG002.jpg ...IMG300.jpg.

- The synchronization file, it contains two column: the first corresponds to the frame index IMGxxx.jpg and the second corresponds to the exposure time.

4.5 Conclusion

We have shown in this chapter the library that I contributed to develop during my PhD. The graphical user interface is also one of the fruits of my participation. We arrived finally to a highly interactive version. However, we want this software to be the base of something bigger, a reference by the neuroscientists for testing statistical models and analyzing their spike trains from MEA acquisitions. Currently, we can imagine many perspective to improve the software at the computational level as well as to improve the user experience. At the computational level, we did a big effort to parallelize the Montecarlo method but still, we believe that we can improve the algorithms and reduce the computation cost by parallelizing other parts of the library. We have also encountered the problem of the very big number of parameters to fit when the networks get larger, which we will discuss in details in the perspective part.

At the user experience level, we consider the current version of *EnaS* as the intermediate between the biologists and the computer scientists. Both of the two communities are interested in interpreting the statistical models in order to discover the network structure and functionalities in the biological neural networks. The current version provides a primordial framework for the ultimate goal. For that, other perspectives are expected at this level, we cite some of them here:

- Adding algorithms to detect the receptive fields and showing the corresponds stimulus for each neuron.
- Creating tools to recognize the type of cells, whether they are OFF or ON. We believe that this type of information may help in analyzing the data.
- Adding tools to make the features selection in order to avoid the increasing number of parameters. The methods we developed in this PhD allow to work with large number of neurons, but the large number of parameters, especially those that are not relevant statistically add bias to the estimation (see the last chapter).

Chapter 5

Analyzing real spike trains with EnaS

Contents

We present in this chapter the application of our method on real data acquired with MEA. Analyzing the MEA data has been a big challenge and it was, actually, more difficult than analyzing data generated synthetically. One of the main reasons is that, in this case, we do not know a priori which potential form to choose. We have seen in chapter 3 that the Maximum Entropy performs well on artificial data. Artificial data have been generated with a known probability distribution. This implies that we know already the parameters and features. In contrary, we have no clue about biological data and we do not have a reference probability distribution to compare to the obtained results. After development of techniques that allow large scale statistics in spatio-temporal scale and testing it on artificial data, our aim is to apply this method on real data in order to figure out the interactions that govern the dynamics in the spike train. This chapter is divided into two main parts. The first part (section 5.1) presents the MEA acquisition from

dissection of the retina until spike sorting, more precisely what I learned from my 3 weeks visit to the **Centro de Neurosciencia de Valparaiso**[1] in Chile, where I participated in each part of the experiments. These experiments have been realized in the context of Keops[2], a Franco-Chilian project (ANR - CONYCIT) where retinal behavior with natural images is of main interest. In the second part (section 5.2), I show the results obtained when applying our Montecarlo estimation method on real data. Finally, we present a conclusion on analyzing real data and the encountered problems.

5.1 Data acquisition and preparation

From retina preparation to data acquisition, there are 5 main steps:

- Designing the stimulus.
- Preparation of the solution.
- Dissection of the retina.
- Calibration and verification of the acquisition system.
- Launching experiments.

5.1.1 Designing the stimulus

We are obliged to respect two constraints related to the stimulus in the experimentation. First, the stimulus should not last for more than a certain time (1 hour in general) because the retinal pigments disappear along the experiment, and since the retina is not in the animal body, no pigment could be regenerated. Second, we have to respect some amount of light in order not to make an over-exposure and damage the tissues. From the last two points, one should choose the stimulus content, time and protocol in an intelligent manner. It has been suggested that stimulus could be given with a high frequency exposure in order to get as much responses samples as we can. However, stimulus frequency varies from one equipment to another. The Chilean team adopted a frequency of 30 fps (Maximum capacity of the provided projector). 120 fps has been used by [Fairhall et al., 2006], but with non-natural image stimuli. For their experiments, they decided to use four different natural stimuli (Figure 5.1 show two sample images from the

[1] http://cinv.uv.cl/en/
[2] https://project.inria.fr/keops/

(a) Sample 1. (b) Sample 2.

Figure 5.1: Two sample images captured using a robot camera that moves like a Degu. The images show a sample of the environment in which the videos were taken.

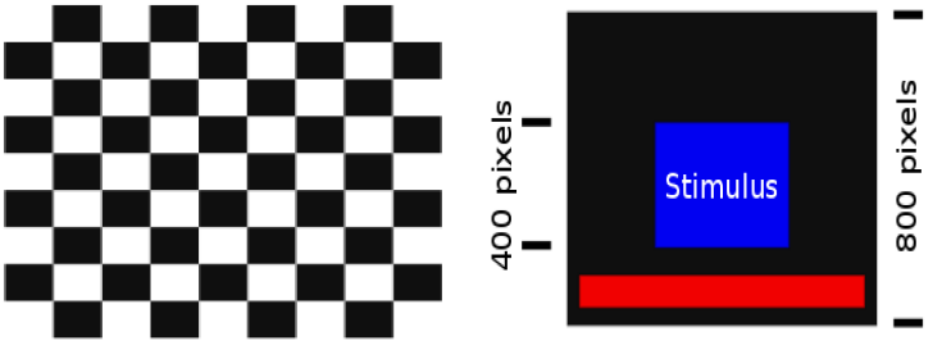
frames environment), taken with a robot camera that walks around on a barren (Simulation of the Degu movement):

- Toward motion (the animal walks in the forward direction).
- Exit motion (the animal walks backward).
- Circular motion (the animal moves its head in a circular trajectory).
- Attention motion (the animal does not move, but it moves only his eyes randomly in the scene). This is an analogue of saccade motion.

Here are the constraints of the experiments, with numbers:

- 20 minutes a synthetic stimulus, contains 50% white and 50% black square, in order to estimate the receptive field (Figure 5.2(a)).
- Stimulus 400×400 pixels size is embedded into an image of 800×800 pixels (Figure 5.2(b)). The image also contains a red horizontal bar, who intensity changes from a frame to another (24 then 64 then 128 then 256 then 24 ...). This code is used as a time stamp in order to know at which time each stimulus was exposed.
- Exposure frequency : 30 fps.

The chronogram (Figure 5.3) of the stimulus begins with 20 minutes of black/white images, where the positions of black/white squares changes along the 20 min, keeping equal the number of white and black squares. The



(a) White-Black square image used to compute the receptive field.

(b) The stimulus log in the exposed image.

Figure 5.2: The stimulus frame.

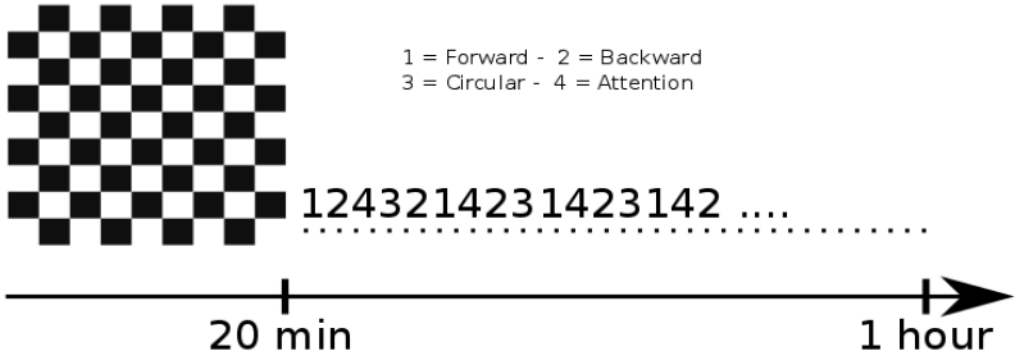


Figure 5.3: The stimulus chronogram. 20 min of black/white squares stimulus followed by 40 minutes of real films stimulus, with ~ 13 repetitions, each, with different order at each repetition.

4 natural stimuli are given afterwards by shuffling randomly the order of the films. Each film is exposed for 15 seconds. A total of 1 hour has been used, which means 60 repetitions for each stimulus.

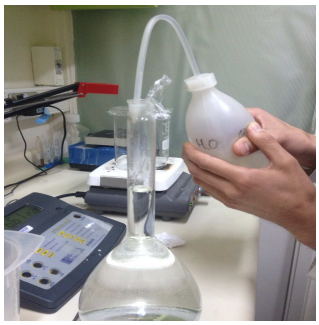
5.1.2 Preparation of the solution

The retina, after isolation from the animal body, needs -to stay alive- to be saved in a quasi-biological medium (solution). The prepared solution will be used during the dissection as well as during the acquisitions. The solution

contains:

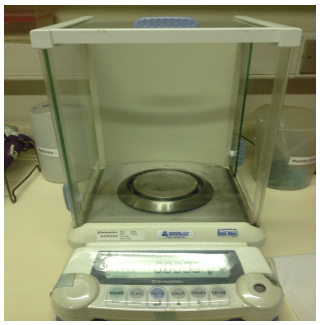
- Sterilized water, Figure 5.4(a).
- Ames Solution (A mixture of different substances, conceived especially for in-vitro experiments), Figure 5.4(b).

We use a low-sensitivity balance in order to get an accurate weight of the given solution (Figure 5.4(c)).



(a) Sterilized water.

(b) The Ames solution.



(c) High-sensitivity balance.

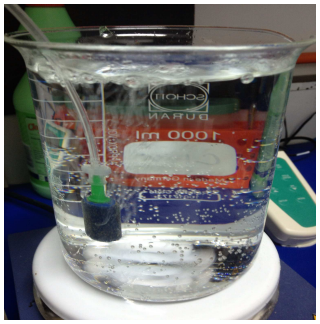
Figure 5.4: First steps in the solution preparation: mixing of sterilized water with Ames Solution.

Once having this mixture, we measure its PH (Acidity level) -with a PH sensor (Figure 5.5(a))- and we regulate its value whether by adding an acidic solution (HCL - Hydrogen chloride) or a basic solution (NAOH - Sodium hydroxide). The PH level should be 7.5.

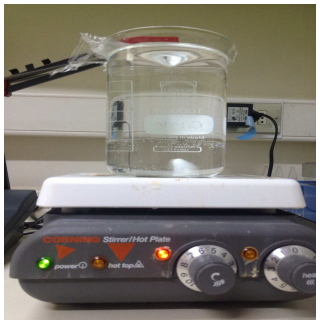
The solution will be saved at a 30 degrees temperature level, using a heater (Figure 5.5(b)).

5.1.3 Dissection of the retina

For the experimentation, we used the Degu (Figure 5.6(a)). It is a mammalian animal whose retina cones are sensitive to blue and green, exclusively. Consequently, this animal does not see in the red light. This advantageous characteristic allows one to make experimentation where the technician can use the red light in order to see and supervise the experiment. During the dissection and the acquisition it is mandatory to use the red light and try to not allow any other light to come inside the experimentation room. The Degu is left for 5 minutes in a little box, where already sprayed anesthetic were injected in order to make the animal fall asleep, completely.



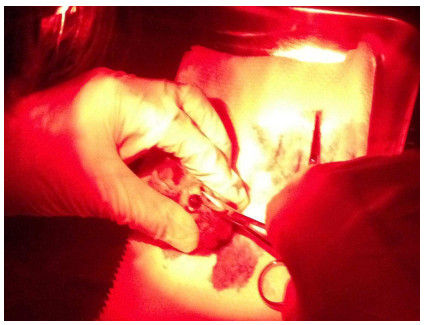
(a) The PH sensor.



(b) The solution heater.

Figure 5.5: Regulating the solution with acid/basic solution and heating until 30 degrees.

(a) The degu (Octodon degus).



(b) Taking out the whole eye from the animal.

Figure 5.6: The degu's eye.

Using a microscope and the dissection tools, one proceeds to the eye extraction, under red light, an ambient temperature of 20 degrees (Figure 5.6(b)).

Once the two eyes are out, one puts each of them in a solution, and keeps an oxygen connection in order to simulate the animal body medium. One now takes one of the two eyes and proceed to the dissection, under the microscope. The dissection of the retina should be done carefully and precisely in order to avoid any injury to the retina tissue:

- The first step is to create a hole at the level of the Sclera (the tissue that separates retina from the rest of the eye cup).
- With a scissors, one begins from the hole to cut all around the eye in order to take out the eye cup.

- One separates retina from the Sclera.
- Both retinas are put afterwards in the biological medium.

Figure 5.7 shows what are the steps followed after the eye extraction.

5.1.4 Calibrating and verification of the acquisition system

Once the retina is ready, one puts it between the MEA and a membrane (technique known as sandwich since the retina is between two parts). Obviously, the ganglion cells should be on the MEA side. The membrane has two main functions. First, it ensures that the retina does not move. Second, it is a pathway where oxygenated solution flows during the experimentation (Figure 5.8(a) and 5.8(b) show respectively the MEA and the membrane, where the white tube carries the oxygenated solution).

The acquisition system

The acquisition system consists of:

- The stimulation computer. It sends the stimulus through electrical connectors to the projectors.
- The optical path, where the stimulus goes through some optics in order to arrive and fit in the MEA.
- The MEA cage (including the MEA itself, the retina and the membrane).
- An analog to digital converter. It performs pre-filtering and amplification, and converts analog signal to digital and sends them to the computer, where data are monitored and saved.
- The display and post processing computer. It displays raw data in real time. It also stocks the acquisition that will be used later for a post-processing on the same computer or another one.

Figure 5.10(a) shows a diagram for the acquisition system. In order to synchronize the stimulus with the acquisition, one takes benefit from the fact that the Degu retina does not perceive the red light and put in each image a red-bar, whose intensity changes with stimulus, as an indicator of the exposure time of each stimulus (Figure 5.9 shows the light separation within the optical path).

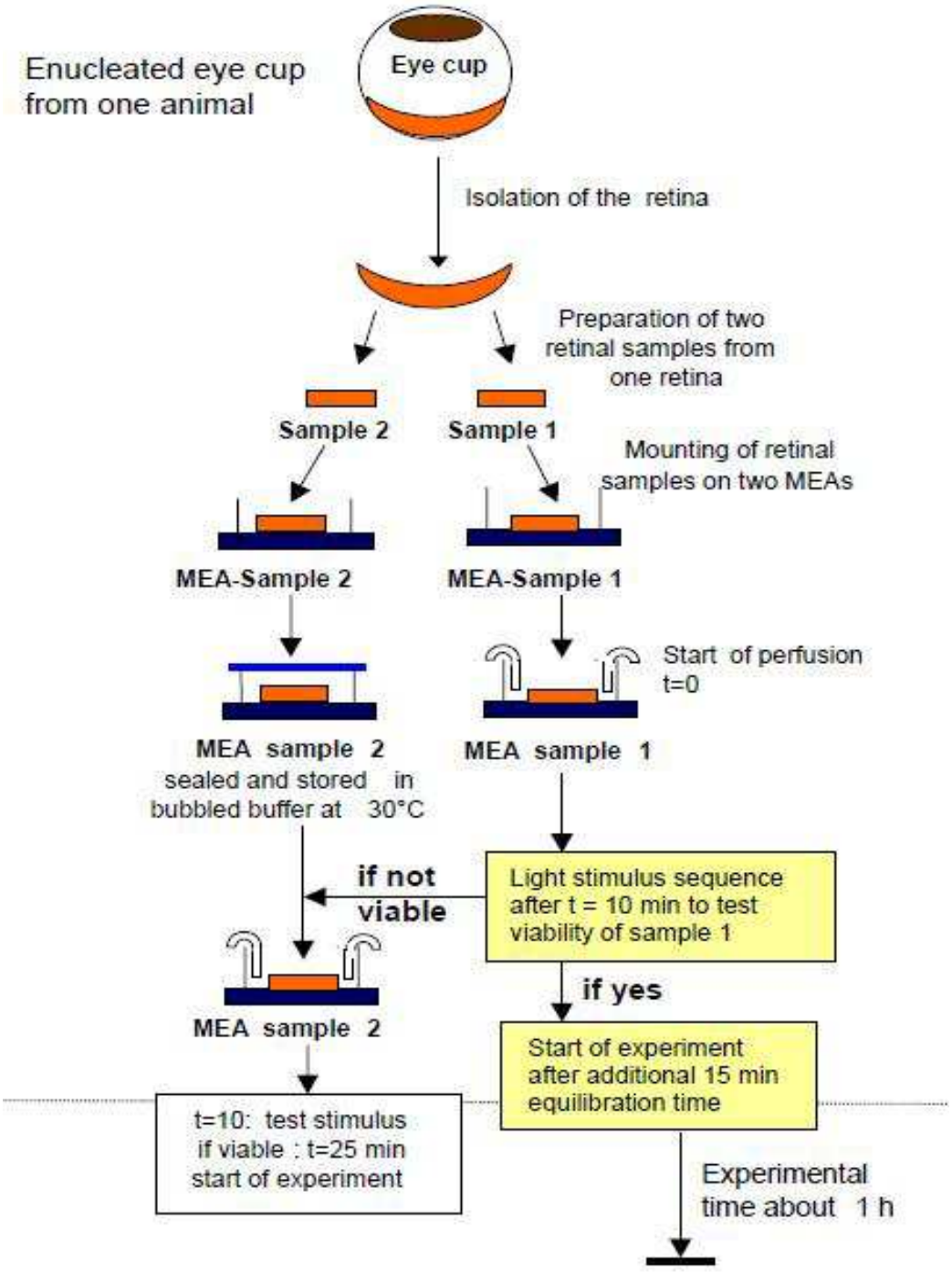
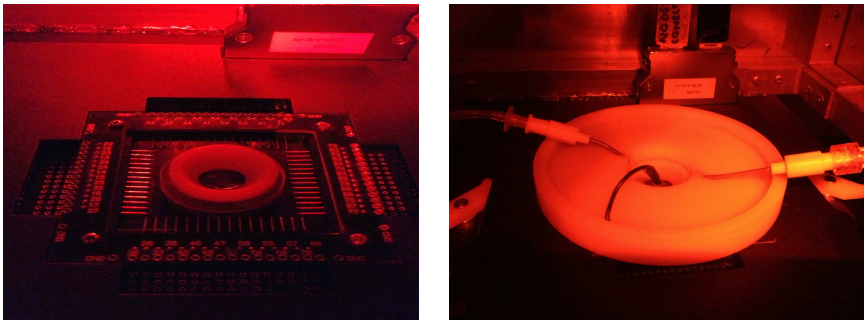


Figure 5.7: Retina dissection road map. Taken from [Multi Channel Systems, 2013]



(a) The multi-electrode array.

(b) The membrane with tubes carrying the oxygenated solution.

Figure 5.8: The multi-electrode array and the membrane.

The MEA cage (Figure 5.11) should be isolated from external noise, as much as possible. Isolation is performed by the aluminum metal around the MEA. This disposition will keep the MEA safe from any external electrostatic noise. Oxygen and solution supplies should be always in the membrane.

Calibration and verification of the acquisition system

Before sending the stimulation frames, it is recommended to verify that 1- the whole system is well setup and 2- the light stimulus that will be exposed at the retinal tissue will not burn the tissue. For the system verification, one monitors the local field potential (LFP) (Figure 5.12), the signal generated by the whole retinal cells. In the same time, one exposes, gradually, a uniform stimulus, in order to see if the LFP is being generated. For the calibration, while increasing linearly the stimulus intensity, one monitors the retinal sensitivity (which is supposed to be a sigmoid function of light intensity of the projector). The used intensity will be the inflection point of the sigmoid function.

5.1.5 Pre-processing and spike sorting

The acquired data contains high and low frequency signals. The low frequency signals corresponds to the local field potential. We are interested in the signals that carry the spikes, those of high frequency. The acquisition system is designed in such a way that we can choose the transmission channels, whether to transmit the high or low frequency signals. This first pre-processing step proceeds the spike sorting.

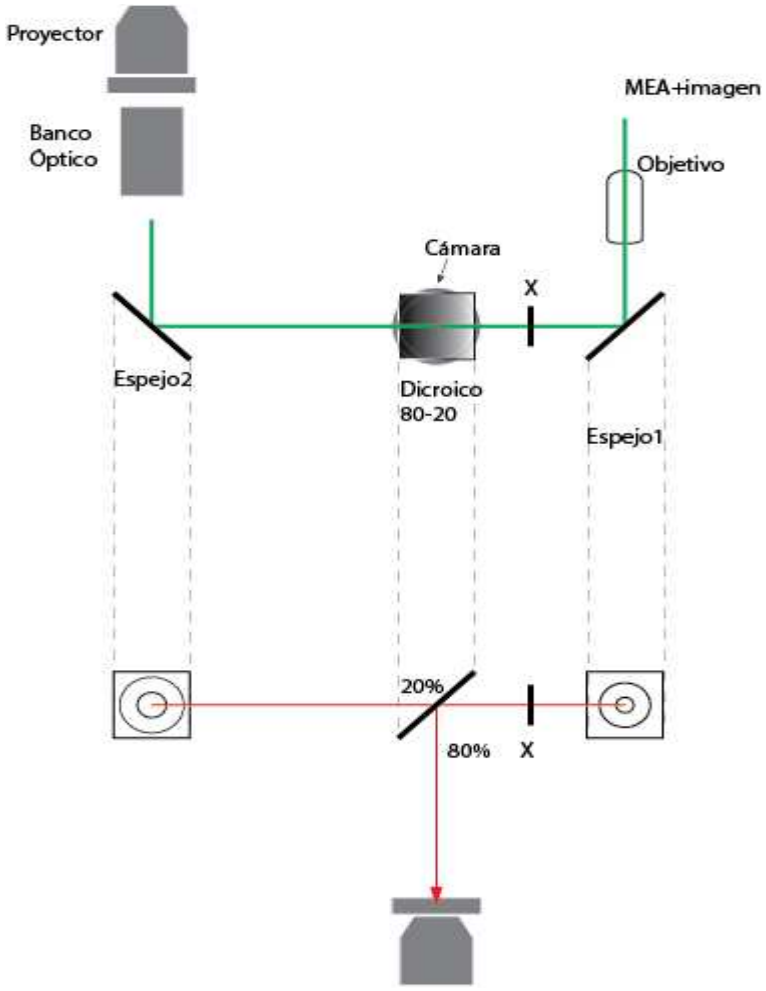
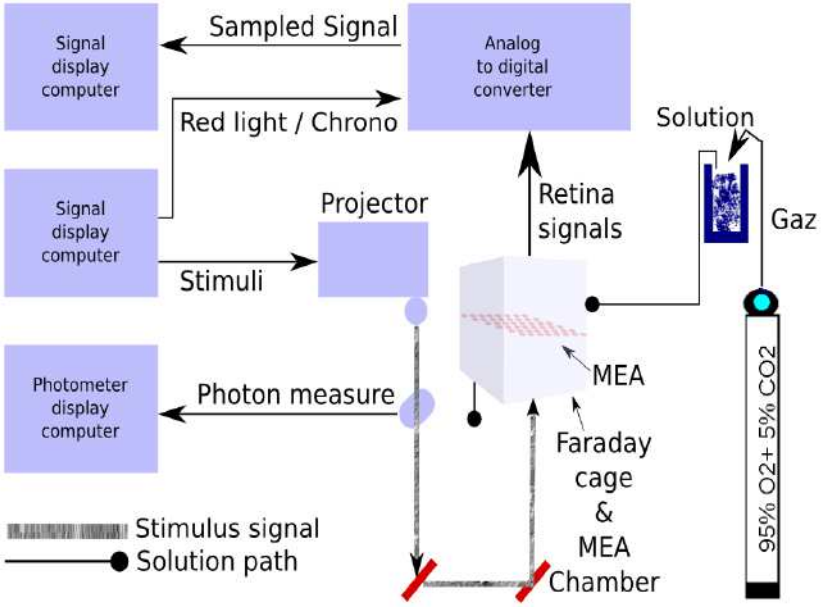
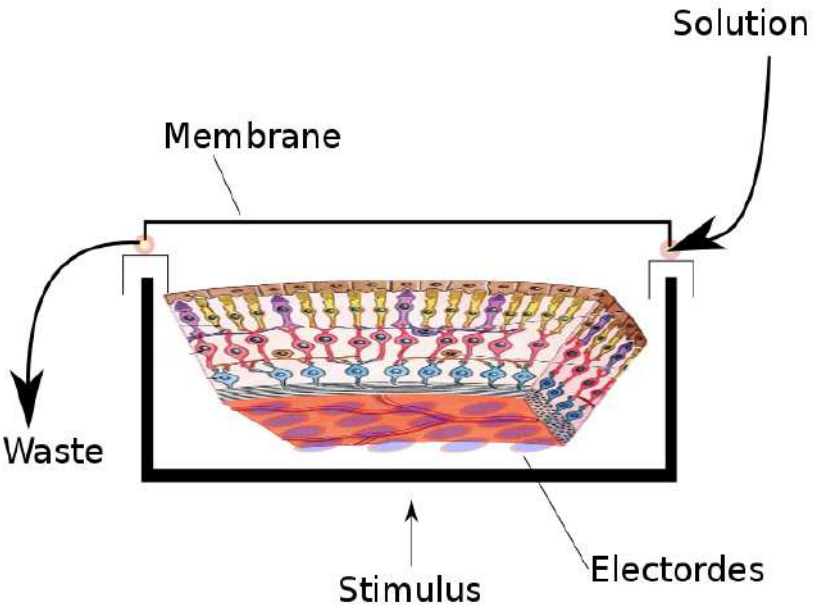


Figure 5.9: The optical pathway. Red light path is separated from the stimulus path.



(a) The block diagram of the acquisition system



(b) The retina between the membrane and the MEA sensor

Figure 5.10: The acquisition system and the MEA sandwich.

Figure 5.11: The MEA cage. A Faraday cage to avoid the electrostatic noise.

Each electrode in the MEA can receive signals from many surrounding cells. Each of the surrounding cells can send signals to many surrounding electrode. The spike sorting consists of determining the number of neurons sensed by each electrode as well as the time stamps of each neuron spikes (Figure 5.13). They performed the spike sorting using the Plexon software.

The spike sorting consists on many signal processing step. Figure 5.14 shows a brief description of those steps. The idea is not to explain the spike sorting in details, but just to give a general view. The spike sorter receives the raw data, that contains high frequency signals from the electrodes. We ensure first the data band by applying a band pass filter [300-3000] Hz. Then, we detect the location of the spikes along the whole signal time window by applying a threshold to the raw filtered data. For the threshold, we use the formula presented in [Quiroga et al., 2004] :

$$thr = 5\sigma_n \qquad \sigma_n = median\{\frac{|x|}{0.6745}\} \tag{5.1}$$

where x is the band-pass filtered signal and σ_n is an estimate of the standard deviation of the background noise. Once the spikes are located, we store all the spike signal (2 ms around the spike peak). For each detected spike, we take the signal from all the other surrounding electrodes, at the same time stamp in order to make the clustering. This data set will allow us to discover what is happening in the surrounding electrodes in the same time window, in order to make the classification.

One uses now those data, for each electrode/spike now, with the independent component analysis (with 5 components) and clusters them using only

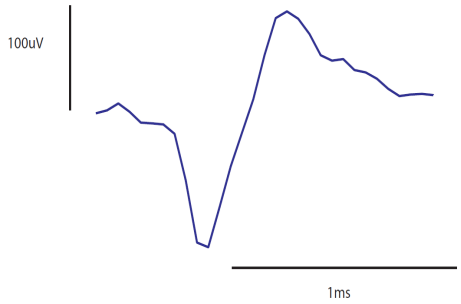


Figure 5.12: The waveform of the local field potential. For the calibration, we must see this signal appearing from the electrodes in order to confirm that we can begin the experiment. One also use the peak of this signal to draw the sigmoid function of the of signal (intensity) in order to get the inflection point.

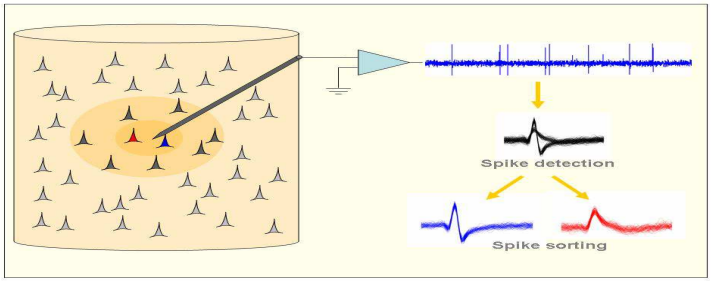


Figure 5.13: Each electrode captures signal from many cells. The electrode signal is amplified and then high-pass filtered

the first two components. For example, Figure 5.15 shows that there are 3 neurons in the cluster.

Now, one will have one or more time stamp vectors for each electrode. Each of them corresponds to a neuron. However, there is a possibility that the same neuron fires at two different electrode and then the same neuron will be represented twice with two different time stamp vectors. Those vectors could be eliminated automatically by detecting a null inter-spike interval or manually by observing the closed time stamp vectors.

Figure 5.16 shows one example of the raster plots obtained at the CINV.

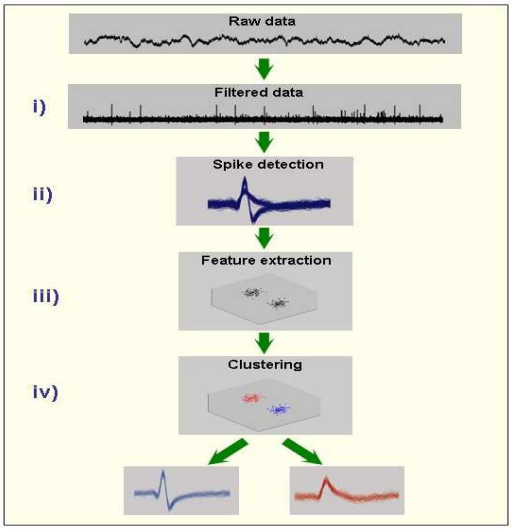


Figure 5.14: A brief description of the spike sorting steps (from [Quiroga et al., 2004]).

5.2 Real data analysis

In the previous section, we discussed the acquisition details. Afterwards, we are interested in analyzing the acquired data. The process consists on:

1. **Suggesting a hypothesis model**. We can choose models that have been used widely in the literature such as pairwise, triplet. quadruplet. Thanks to *EnaS*, we can use one of these models as well as model with spatio-temporal constraints. Furthermore, we can edit our own model.

2. **Parameters fitting**. This step that has been explained in chapter 3 aims to find the parameters that ensure that the model is as close as possible to the data (the one with the minimum Kullback-Leibler divergence).

3. **Model evaluation**. It consists of evaluating how close is the hypothesized model to data. For that, we use quantitative tools that allows us to evaluate the error committed using such model. Those tools are: Kullback-Leibler divergence and confidence plots (for details about the confidence plots and KLD, refer to the section 4.4.3).

The data set we used is a spike train of 40 neurons (Figure 5.17) provided by one of our collaborators: Olivier Marre and Michael Berry from the university of Princeton. This data set has been studied before in [Schneidman et al., 2006]

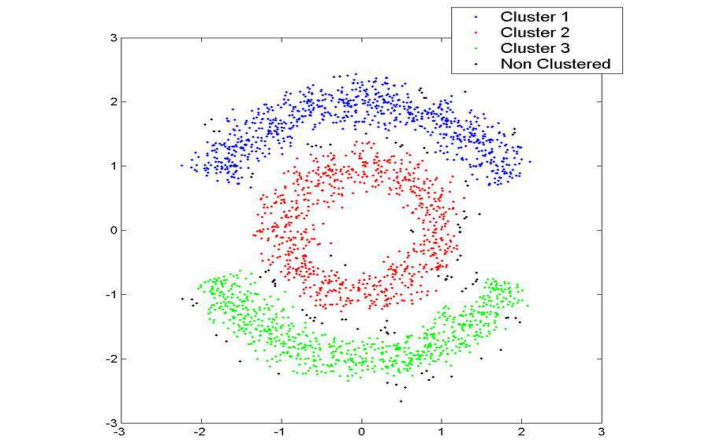


Figure 5.15: Clustering of two main components of the ICA, three cells are resulting. Taken from [Quiroga et al., 2004].

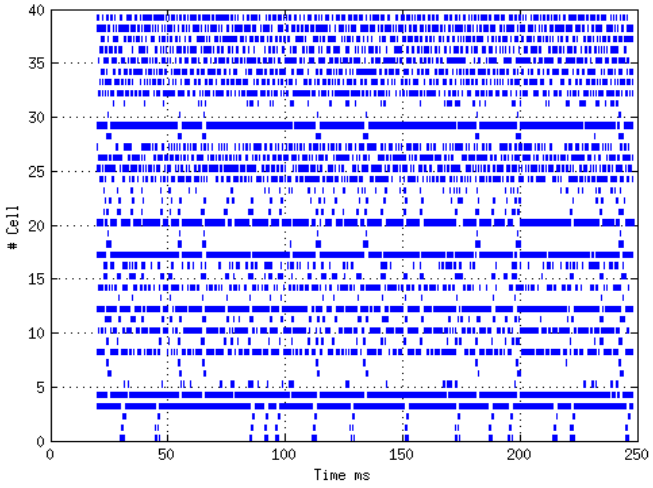


Figure 5.16: An example of the raster plots obtained at the CINV. This spike train has been spike sorted from retinal acquisition on the Degu. The retina was stimulated by a movie of natural images.

but on 10 neurons only for fitting an Ising model. We fitted an Ising and a pairwise model of range 2 on 20 and 40 neurons using this data set. We show (in the coming figures) the comparison of monomials and patterns probabilities (observed and predicted) in both cases. We also show an example of error (Hellinger distance) evolving during fitting the parameters in the case of pairwise $R = 2$. The data is binned at 20 ms similarly to

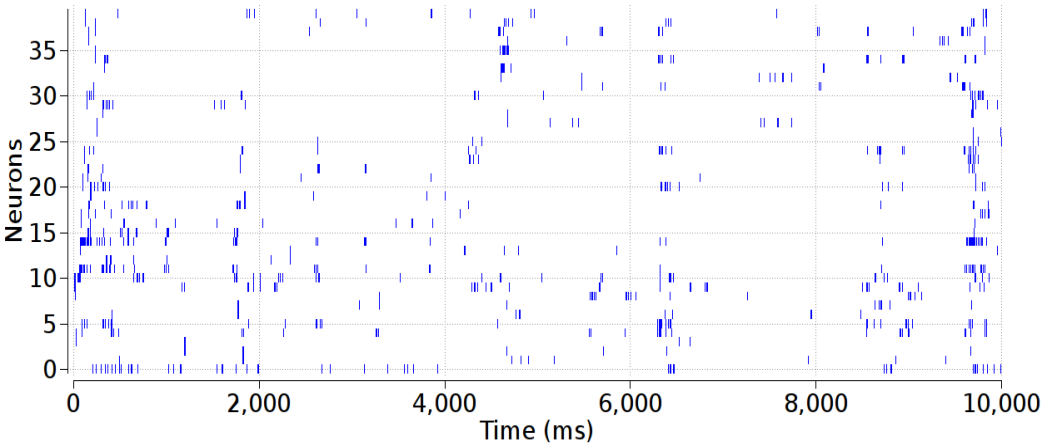
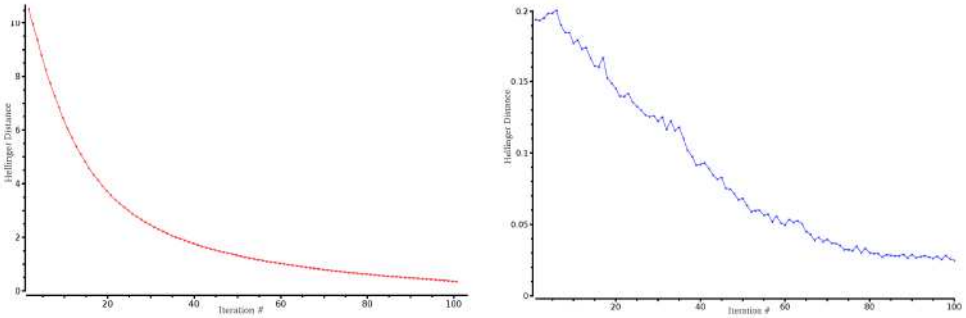


Figure 5.17: A set of 40 neurons acquisitions on salamader retina. Spike train length = 3179551 ms. Courtesy: Olivier Marre.

[Schneidman et al., 2006].

Figure 5.18 shows the evolution of the Hellinger distance during the parameter update both in the parallel and sequential update process.

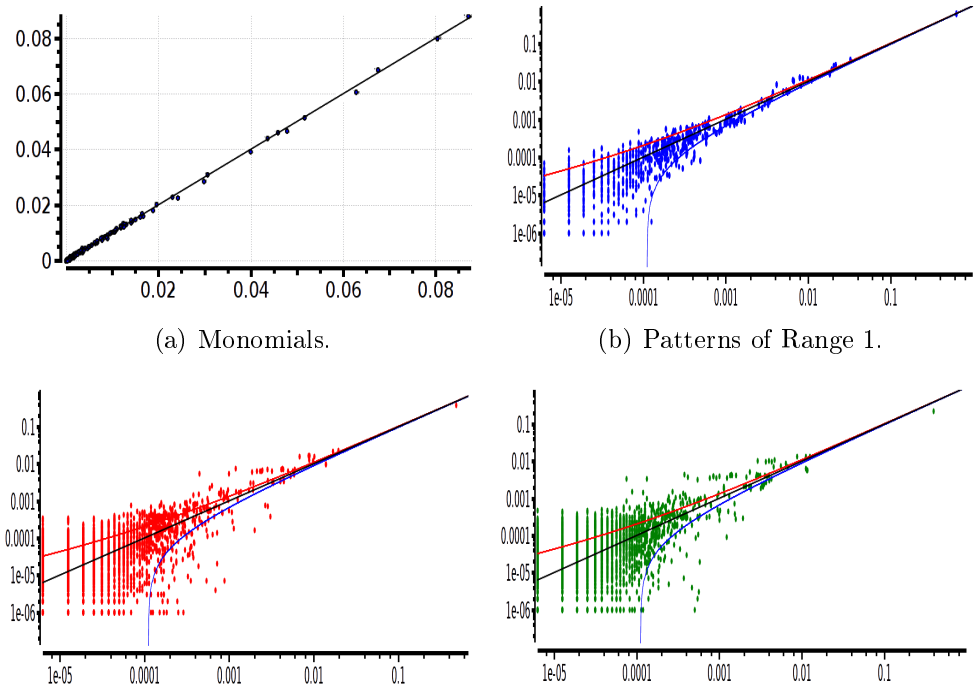
After estimating the parameters of an Ising and pairwise model of range $R = 2$ on a set of 20 neurons, we evaluate the confidence plots. Figures 5.19 and 5.20 show, respectively, the confidence plots for patterns of Ranges 1, 2 and 3 after fitting with an Ising model and the pairwise model of range $R = 2$. Our results on 20 neurons confirm the observations made in [Vasquez et al., 2012] for $N = 5, R = 2$: a pairwise model with memory performs quite better than an Ising model to explain spatio-temporal patterns.



(a) Convergence during the parallel update.

(b) Convergence during the parallel update.

Figure 5.18: Evolution of the Hellinger distance during the parallel (**a**) and the sequential (**b**) update in the case of modeling a real data set with a pairwise model of range $R = 2$. The parallel update provides a fast convergence; however, it is steady after a hundred iterations. The,n we iterate the sequential algorithm.



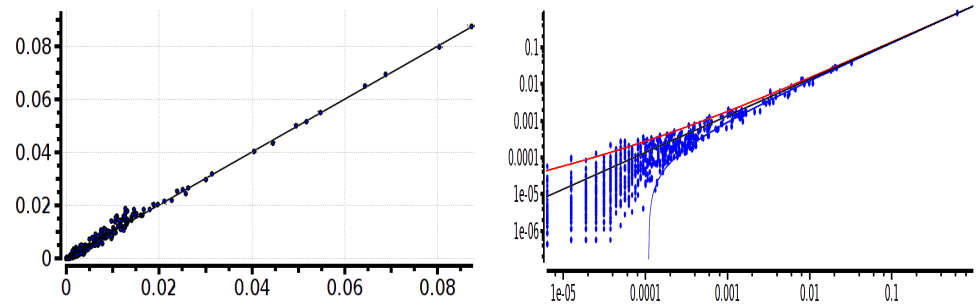
(a) Monomials.

(b) Patterns of Range 1.

(c) Patterns of Range 2.

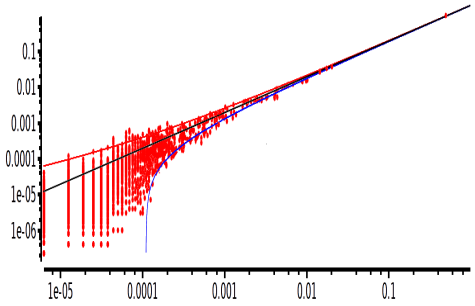
(d) Patterns of Range 3.

Figure 5.19: A 20-neuron data set binned at 20 ms with an Ising model. After fitting, we show the comparison between observed (in the real spike train) and predicted average values of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities for patterns of Ranges 1, 2 and 3, respectively. In (a), (b), (c) and (d), the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The computation time is equal to 18 hours on a small cluster of 64 processors (around 5 min per iteration). The estimated Kullback–Leibler divergence is 0.307.

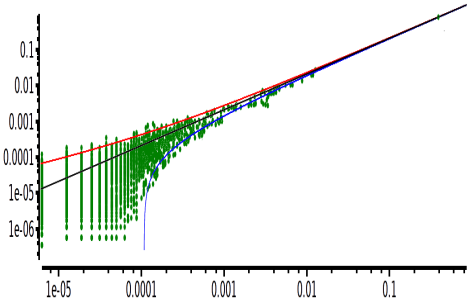


(a) Monomial averages.

(b) Patterns of Range 1.



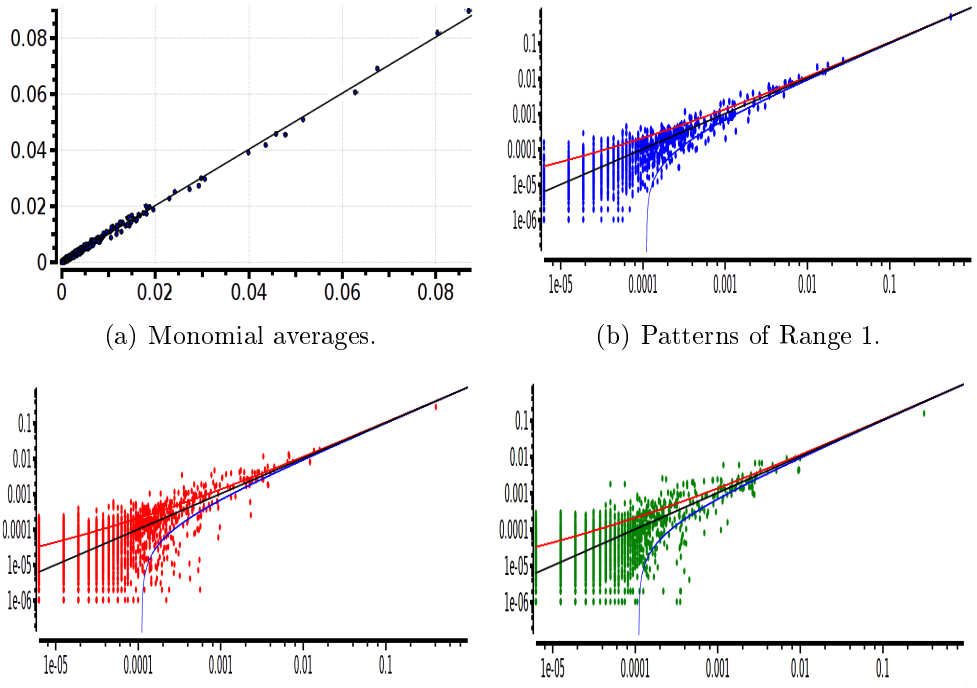
(c) Patterns of Range 2.



(d) Patterns of Range 3.

Figure 5.20: A 20-neuron data set binned at 20 ms with a pairwise model of Range 2. After fitting, we show the comparison between observed (in the real spike train) and predicted average values of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities for patterns of Ranges 1, 2 and 3, respectively. In (a), (b), (c) and (d), the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The computation time is equal to 40 hours on a small cluster of 64 processors (around 12 min per iteration). The estimated Kullback–Leibler divergence is 0.281.

We then made the same analysis for 40 neuron. Figures 5.21 and 5.22 show, respectively, the confidence plots for patterns of Ranges 1, 2 and 3 after fitting with an Ising model and the pairwise model of range $R = 2$. In this case, we were not able to obtain a good convergence for $N = 40, R = 2$. This is presumably due to the insufficient length of the data set, which does not allow us to estimate accurately the probability of some monomials. This aspect is discussed in the next section.



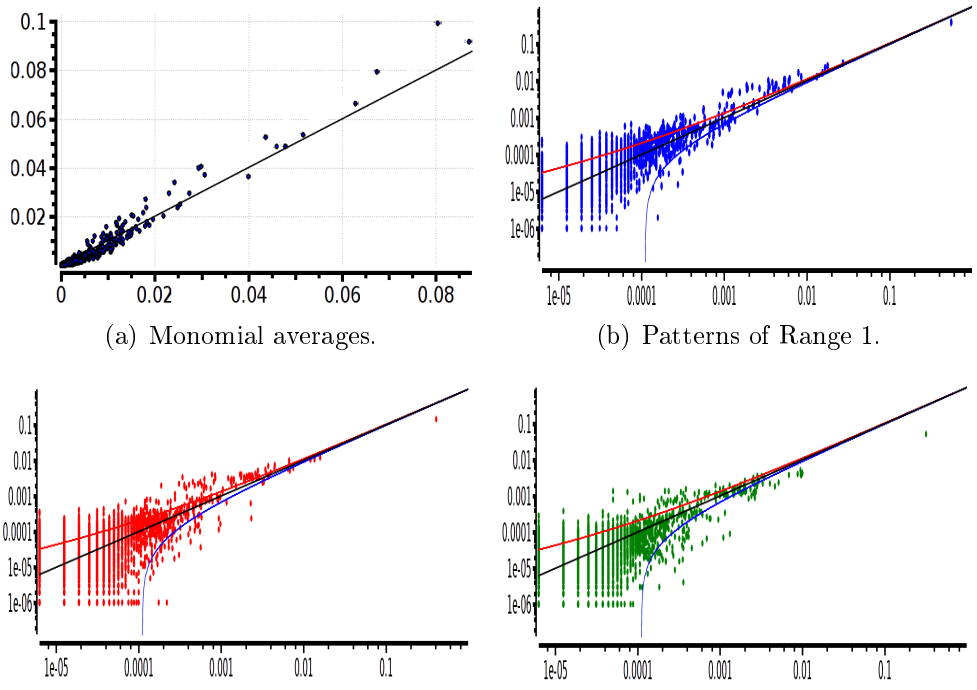
(a) Monomial averages.

(b) Patterns of Range 1.

(c) Patterns of Range 2.

(d) Patterns of Range 3.

Figure 5.21: A 40-neuron data set binned at 20 ms with an Ising model. After fitting, we show the comparison between observed (in the real spike train) and predicted average values of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities for patterns of Ranges 1, 2 and 3, respectively. In (a), (b), (c) and (d), the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The computation time is equal to three days on a small cluster of 64 processors (around 21 min per iteration). The estimated Kullback–Leibler divergence is 0.930.



(a) Monomial averages.

(b) Patterns of Range 1.

(c) Patterns of Range 2.

(d) Patterns of Range 3.

Figure 5.22: A 40-neuron data set binned at 20 ms with a pairwise model of Range 2. After fitting, we show the comparison between observed (in the real spike train) and predicted average values of monomials in (**a**). (**b**,**c**,**d**) The comparison of predicted and observed probabilities for patterns of Ranges 1, 2 and 3, respectively. In (a), (b), (c) and (d), the x-axis represents the observed probabilities and the y-axis the predicted probabilities. The computation time is equal to seven days on a small cluster of 64 processors (around 47 min per iteration). The estimated Kullback–Leibler divergence is 0.983.

5.3 Conclusion

We gave in this chapter an idea about the workflow from retinal dissection until fitting Maximum Entropy models passing through acquisition and spike sorting. The experiments with the Degu animal were realized in the Centro de Neurosciencia de Valparaiso under the supervision of Pr. Adrián Palacios. Participating to those experiments was enriching and it showed how the process is complicated and when we analyzing real data, we are not only analyzing data with unknown model shape, but also with noise that comes from acquisitions, inaccuracy in the measurement and spike sorting.

Concerning analyzing the real data we encountered a problem with two

faces: the big number of parameters L. The problem of having thousands of parameters to fit in a distribution is big obstacle when it comes to real data. We will discuss this problem from several angles:

- The big computational time we are still facing. As we showed in the previous chapter that the computational time increases in a power law with the increasing number of models parameters. For instance, on a cluster of 64 processors, one needs 1 day to fit an Ising model and 2 days to fit a pairwise model.

- The risk of divergence: since we are analyzing canonical form potential, we have an a priori set of monomials in the model. Some of the monomials do not appear in the empirical distribution which affects the convergence of the fitting process. Those monomials do not only cause an error on themselves, but also on the estimation of other parameters. This issue is discussed in details in the last chapter. To circumvent this problem, we imposed a coefficient value equal to $-\infty$ for monomials that appear less than 2 times in the spike train. This issue allowed the process to consider literately that those monomials does not appear in the data which is convenient with the Maximum Entropy paradigm (increasing knowledge about the data).

- The model choice issue: this is a question of concept. If we know that some data follow an Ising law and we fit the model to the data only to estimate the parameters, fitting process goes very nice. However, real data where generated with an unknown distribution which we believe that it is more complicated at the interactions level (non canonical) and contains less parameters that we impose with the canonical forms.

From the last issues, we conclude that analyzing real data with at least a known model form is better and will be much helpful to make us understand the data, save computation time, guarantee the problem convergence.

We have also shown that data at a large scale are not distributed as an Ising model. This model allows to predict patterns of depth 1, however, it cannot predict patterns of depth 2.

One of the most challenging task for analyzing data at a large scale was tuning the variables of the simulation, especially k and the number of needed iterations. If one would use *EnaS* to compute the parameters, we recommend the following trade-off:

- After choosing the model, compute 100 parallel iterations with $k = 5$ for an Ising model and $k = 10$ for a triplets or spatio-temporal model.

This will make the solution converge faster. We know that the bigger the k the better the convergence. However, in the first iterations, k is not that sensitive.

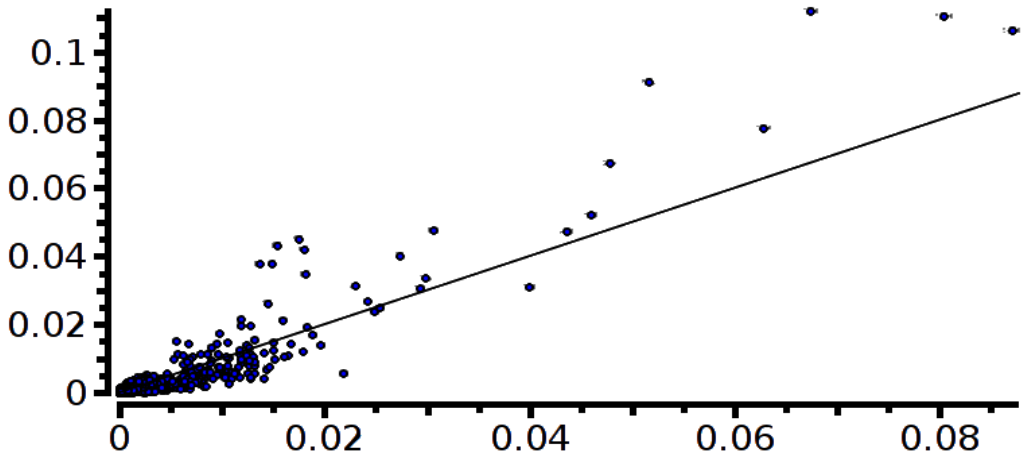
- Afterwards, if you see that the monomials averages are still far to be aligned on the the equality axis, run more 50 parallel iterations. To give an idea of what we consider "far to be aligned", see Figure 5.23(a). If you see that the parallel algorithm gave somewhat good estimation (for example as the one shown in 5.23(b)) on the monomials, you can go to the next step.

- Before proceeding to the next step, it is important to note the following remark: in the implementation of the algorithm, we allow more relaxation in the parallel update that in the sequential update. For instance, the error we allow on estimating the monomials is smaller when the sequential algorithm is run. The reason is that we realized that the parallel update algorithm does not allow very small errors on the estimation since it is updating all the parameters at once. The sequential algorithm allows to do so because it is updating one parameter at once.

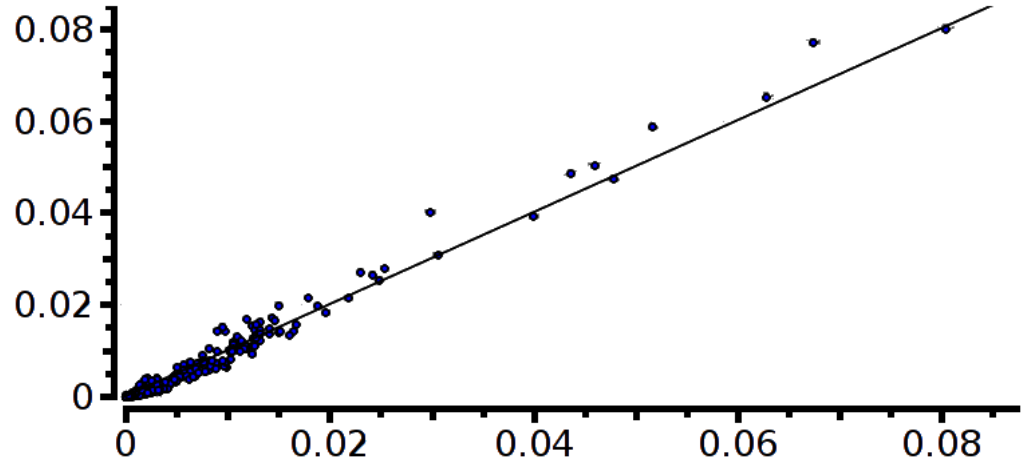
- Now, proceed to the sequential iterations. Choose $N_{flipp} = 30$ and run 200 iterations. Normally, this amount of iterations is enough. You might need to do more, but not more that 2 runs of 50 iterations.

- Once you arrive to a good convergence, for instance Figure 5.23, you can compute the Kullback-Leibler divergence and the confidence plot. At this stage, one must choose a big k, for instance 50-100 in order to compute a Montecarlo raster that represent at best the estimated parameters. On this Montecarlo spike train, *EnaS* computes the KLD and the predicted patterns probabilities.

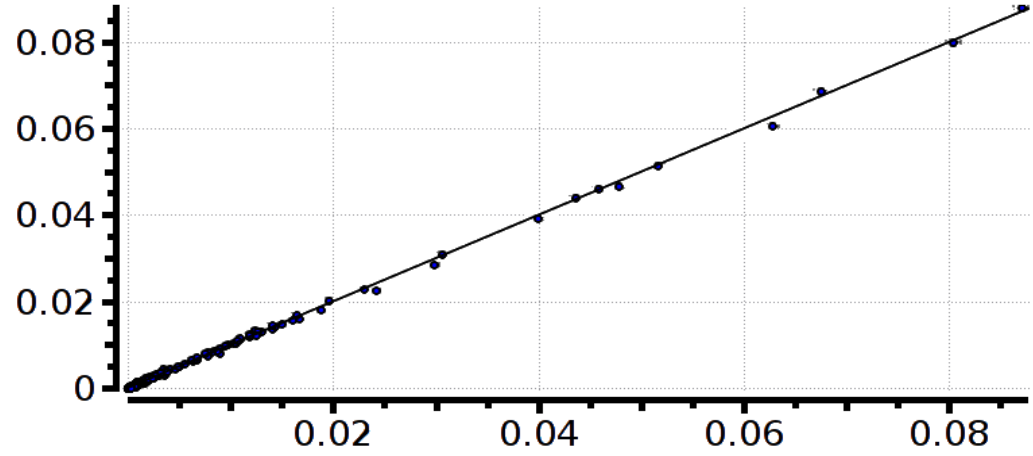
- It is preferred to run this process of a cluster of at least 16 processors for 10-20 neurons and 64 processors for more than 30-40 neurons.



(a) Far to be aligned



(b) Almost aligned



(c) Good convergence

Figure 5.23: An example of monomials averages after parallel update. (a) presents the case where we still need more iterations to converge and (b) presents the case where we stop the parallel algorithm and go for the sequential updates. (c) presents and example of the monomials averages comparison after the convergence.

Chapter 6

Conclusion and perspectives

Contents

A number of questions raised up during this work, especially concerning fitting the parameters and the advantages of our Montecarlo framework with respect to existing framework. We will uncover those questions in this chapter, discuss them and suggest road maps for possible upcoming perspectives.

6.1 On the binning of the spike trains

The binning (see Figure 4.5 for details) is an influential parameter in analyzing the spike trains and indeed, it should be taken into account rigorously. The fact that a 10 or 20 time steps of the activity of a neuron (in the spike train before binning) is replaced by 1 time step (in the spike train after

binning) is, definitely, a loss of information. In the other counterpart, researchers have always used binned spike trains to fit Maximum Entropy models ([Schneidman et al., 2006], [Ganmor et al., 2011a], [Vasquez et al., 2012]).

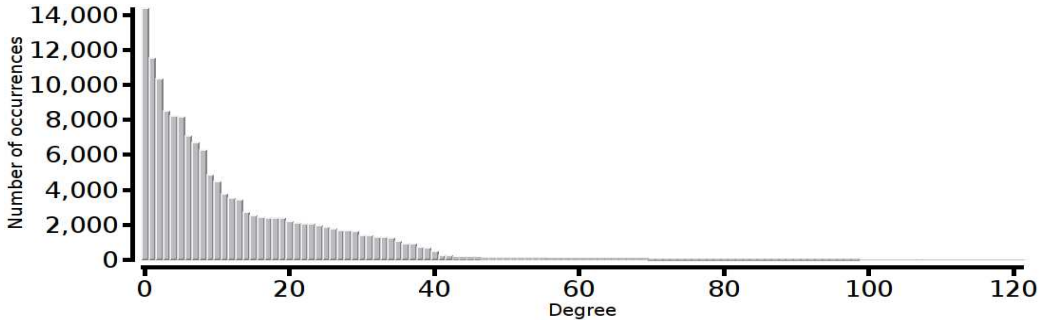
An example of the empirical statistical changes that could result from binning is presented in the Figure 6.1. The test is applied on the data set D_2. We plot the histogram of patterns in the data where we bin at 1, 5, 10 and 20. Figure 6.1 shows that as we increase the binning, more patterns appear in the spike train. For instance, when the binning is 20, the number of pattern increase by 700% than when binning is 1!

The change in the empirical distribution due to binning implies the change of the whole set of constraints. The Maximum Entropy Principle that is investigated here is based on constraining the monomials. By consequence, we are finally studying the binned activity of the neural ensemble and not the activity at their biological time scale. One advantage that may result from binning data is that, when using canonical form potentials, we have less monomials with empirical probability equal to 0 and so we have less risk for the solution to diverge (we explain in the next section why non-appearing monomials affects the estimation). However, in the same time, we might be adding monomials that do not exist in the spike train. Also another counterpart, binning at 1 may result shedding the canonical form of potential because we will have more non-appearing monomials than appearing ones, especially for higher order models. The solution that may overcome this kind of problematic is a features selection method, which we discuss in the next section.

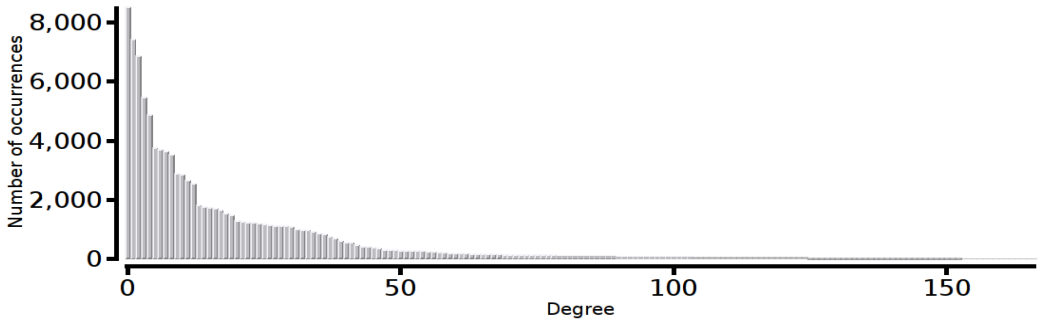
6.2 On the non-existing canonical potential monomials

With a set of 40 neurons, relatively small compared to the new technologies of MEA systems, we find ourselves fitting models with thousands of parameters for dozens of neurons. For instance, as Figure 6.2 shows, for 40 neurons the number of canonical potential parameters (L) in the case of the following models is:

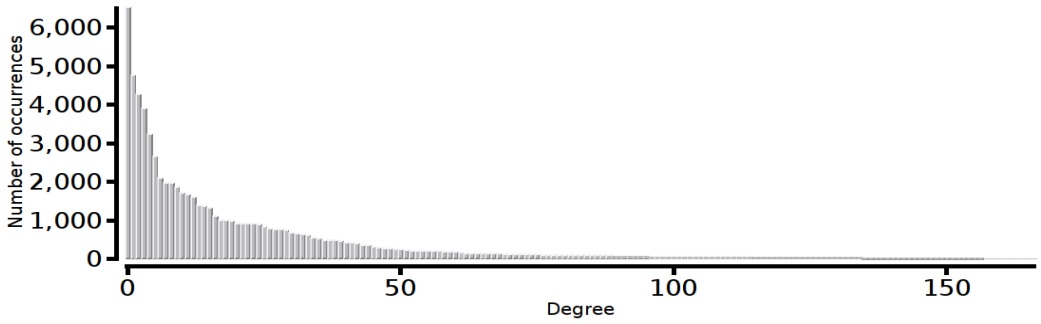
- $L_{Ising} = N + \frac{N \times (N-1)}{2} = 820$
- $L_{Triplets} = N + \frac{N \times (N-1)}{2} + \frac{N \times (N-1) \times (N-2)}{6} = 10700$
- Pairwise with memory D: $L_{Triplets} = N + \frac{N \times (N-1)}{2} + (R-1) \times N \times (N-1)$
    1. For $R = 2$, we have $L = 2380$ parameters.



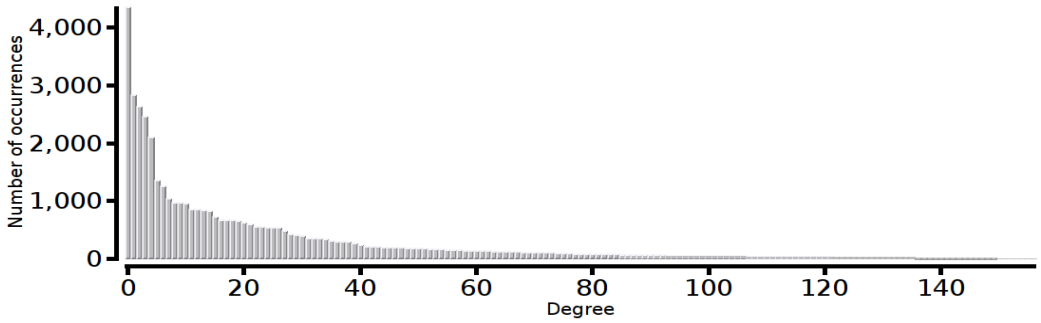
(a) Binning = 1 - Number of found patterns = 1324



(b) Binning = 5 - Number of found patterns = 4900



(c) Binning = 10 - Number of found patterns = 7725



(d) Binning = 20 - Number of found patterns = 10049

Figure 6.1: The changes in empirical estimation of patterns histogram in term of binning. The degree is the number of firing neurons found in the pattern.

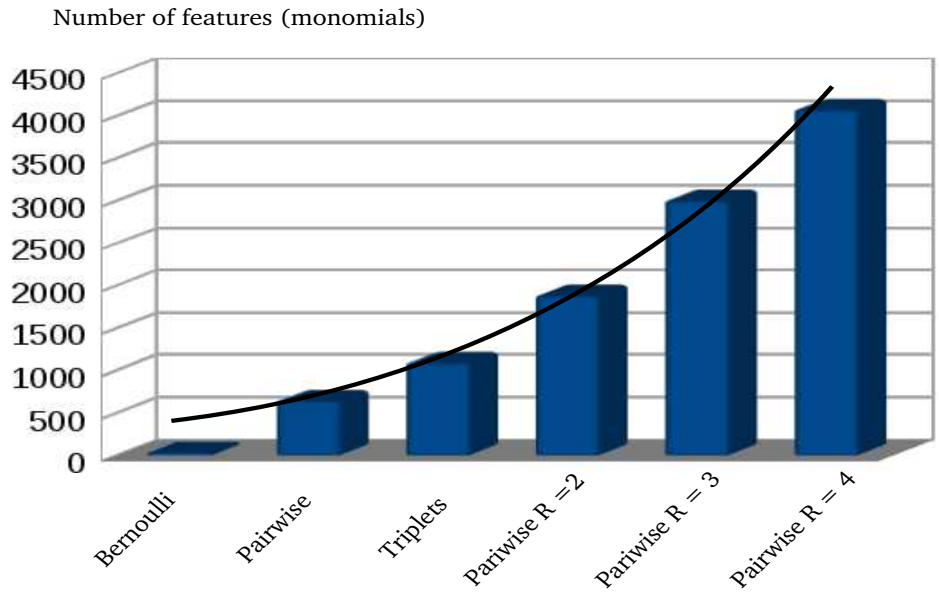


Figure 6.2: The number of parameters dramatically increase whenever the model is of higher dimensions.

2. For $R = 3$, we have $L = 3940$ parameters.

It has definitely no sense to represent such data sets with this number of parameters. Furthermore, this may imply huge errors during the fitting process if we use canonical form potential without any filtering. This error comes from the fact that some monomials, are not only non useful for the models, but also affect the estimation of other monomials in the distribution.

In the case of Maximum Entropy the situation can be described as follows. We generate a finite raster ω_0^T from a known distribution $\mu_{\boldsymbol{\lambda}^*}$ with a potential of the form 2.4. Denote $\mu_{\boldsymbol{\lambda}^*}[\mathbf{m}]$ the vector with entries $\mu_{\boldsymbol{\lambda}^*}[m_l]$ and $\pi_\omega^{(T)}[\mathbf{m}]$ the vector with entries $\pi_\omega^{(T)}[m_l]$. From(2.38) we have $\mu_{\boldsymbol{\lambda}^*}[\mathbf{m}] = \nabla_{\boldsymbol{\lambda}^*}\mathcal{P}$. This exact solution is obtained when the Gibbs distribution $\mu_{\boldsymbol{\lambda}^*}$ can be exactly sampled, namely, for an infinite raster. For a finite raster, if T is large enough to apply the central limit theorem, the empirical distribution $\pi_\omega^{(T)}[\mathbf{m}]$ is Gaussian with mean $\mu_{\boldsymbol{\lambda}}[\mathbf{m}]$ and covariance $\frac{1}{T}\chi$ given by (3.36). We have therefore $\pi_\omega^{(T)}[\mathbf{m}] = \mu_{\boldsymbol{\lambda}^*}[\mathbf{m}] + \boldsymbol{\eta}$ where $\boldsymbol{\eta}$ is centered Gaussian with covariance $\frac{1}{T}\chi$. Solving (2.38) where the exact probability $\mu_{\boldsymbol{\lambda}^*}$ is replaced by the empirical one $\pi_\omega^{(T)}$, one obtains an approximate solution of $\boldsymbol{\lambda}$, $\boldsymbol{\lambda}^*$ with : $\boldsymbol{\lambda} = \boldsymbol{\lambda}^* + \boldsymbol{\epsilon}$, where $\nabla_{\boldsymbol{\lambda}}\mathcal{P} = \pi_\omega^{(T)}[\mathbf{m}]$. Therefore,

$\nabla_{\boldsymbol{\lambda}}\mathcal{P} = \mu_{\boldsymbol{\lambda}^*}[\,\mathbf{m}\,] + \boldsymbol{\eta} = \nabla_{\boldsymbol{\lambda}^*+\boldsymbol{\epsilon}}\mathcal{P} = \nabla_{\boldsymbol{\lambda}^*}\mathcal{P} + \boldsymbol{\epsilon}\chi + O(\|\boldsymbol{\epsilon}\|^2)$. Hence, $\boldsymbol{\epsilon} = \chi^{-1}\eta$. χ is invertible since $\mathcal{P}$ is convex.

The fluctuations of the estimated solution $\boldsymbol{\lambda}$ around the exact solution $\boldsymbol{\lambda}^*$ are therefore Gaussian, centered, with covariance $\mathbb{E}[\,\epsilon.\tilde{\epsilon}\,] = \mathbb{E}[\,\chi^{-1}.\boldsymbol{\eta}.\tilde{\boldsymbol{\eta}}.\tilde{\chi}^{-1}\,]$. Since χ is symmetric we have $\mathbb{E}[\,\epsilon.\tilde{\epsilon}\,] = \chi^{-1}.\mathbb{E}[\,\boldsymbol{\eta}.\tilde{\boldsymbol{\eta}}\,].\chi^{-1} = \frac{1}{T}\chi^{-1}$. We arrive therefore at the conclusion that the fluctuations on the estimated coefficients $\boldsymbol{\lambda}$ are highly constrained by the convexity of the pressure, as expected. Mathematically, everything goes nicely since $\mathcal{P}$ is convex. However, when we estimate the solution numerically, it may happen that $\mathcal{P}$ is quite flat in some directions/monomials. This arises a fortiori since we are not handling the exact values but estimations. Therefore, when considering potentials of the form (2.4) it is expected that some terms (monomials) not only are irrelevant, but also dramatically deteriorate the estimation problem, introducing almost zero eigenvalues in χ.

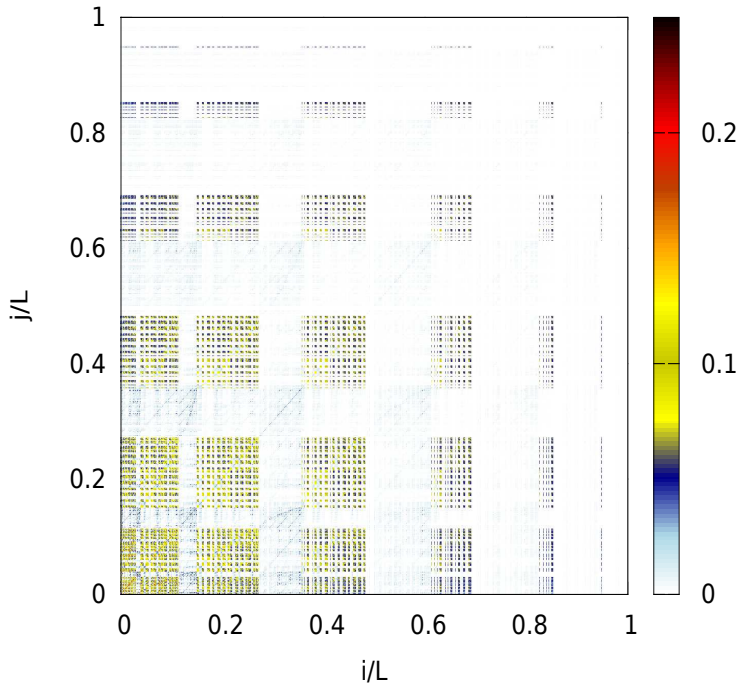


Figure 6.3: The correlation matrix χ, computed using Eq. 3.36. The color bar at right shows the color-value map of χ. Each point in the matrix represents one χ_{ij} entry.

In order to validate this point of view, we have computed the spectrum of χ for two classes of potentials, dense and sparse (explained in 3.7.1), with

5 neurons and $R = 2$ .To analyze the typical spectrum of χ we generate 1000 potentials of the two types. Although such potential has only NR non zero monomial coefficients we consider it as a potential with $d = 2^{NR} - 2^{N(R-1)}$ canonical entries where most coefficients are 0. Thus, χ has a dimension d. For each of these potentials we compute χ and its spectrum. Then, we average the matrix χ over the 1000 samples as well as the spectrum. The result are plotted in Figure 6.3. This figure shows that a big percentage of this matrix is very small or even zero.

6.2.1 Is there any possible solution?

In order to answer this question, we have to ask the following question first: Is it useful to have a model with 4000 parameters for 40 neurons? We believe that this has completely no sense. Indeed, it is normal to get such number of monomial because we want to represent more neurons and higher order models. Thus, a feature selection method is useful and should complement this work. There are many direction we can take in the favor of the monomials selection. For instance, selecting the monomials on threshold ([Rosenfeld et al., 1994], [Koeling, 2000]), using a χ^2 method ([Chen and Rosenfeld, 1999]) as well as incremental feature selection algorithm ([Berger et al., 1996], [Zhou and Wu, 2003]) Other methods based on periodic orbit sampling ([Cessac and Cofre, 2013]) and information geometry ([Nakahara and Amari, 2001], [Amari, 2001]) would be very useful to improve the fitting method.

Another way to make a feature selection is to adapt this model to information we know already about the retina from recent work that tried to discover the functional architecture of the retina. For instance, from [Bloomfield and Völgyi, 2009], we know a priori that neighbor neurons are more likely to fire with a small delay and distant neurons with a bigger delay. This give us a set of constraints that we can impose on the canonical form in order to reduce their complexity.

6.3 What to do after fitting

This question is very important and it plays a big role in the valorization of any work in probabilistic modeling for neural data. How we can invest the parameters we found from the fitting process? Let us assume here the perfect case: we have a data set and we have the pairwise model with a range $R = 2$ that fits perfectly to those data. We can conclude that:

- Instantaneous pairwise interactions and pairwise of memory 1 and 2 time step are essential in the data.

- Other interactions such as triplets and pairwise within more than 2 time steps are useless.
- From the parameters values, we can know what is the statistical influence of each monomial. For instance, the greater the parameter value (in general) the more the monomial is statistically influencing in the data.

The question is now: what can we do with those parameters? This question could be answered in two parts.

- The parameters are a tool and not an aim: this means that once we obtain the parameters, we are only interested in computing the Kullback-Leibler divergence, making confidence plots and enjoy the scientific truth behind the statistical model of retinal data. However, this is not enough.
- For that, the parameters should be an aim and not a tool: not only use them to compute the KLD and the confidence plots but also use them for applications.

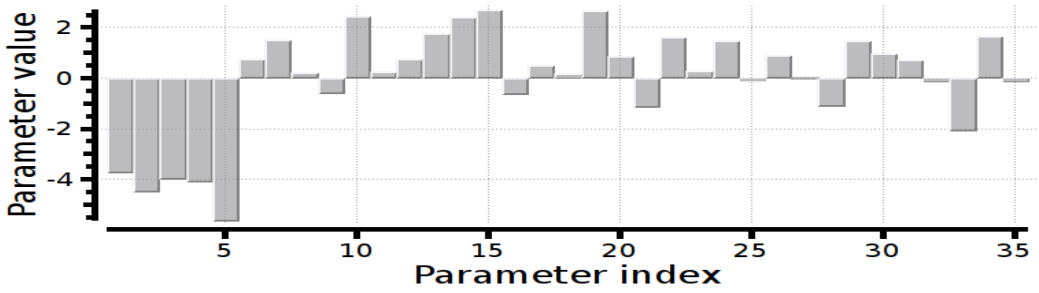
We propose now two suggestions at this level.

Mixing the Maximum Entropy with GLM-like models

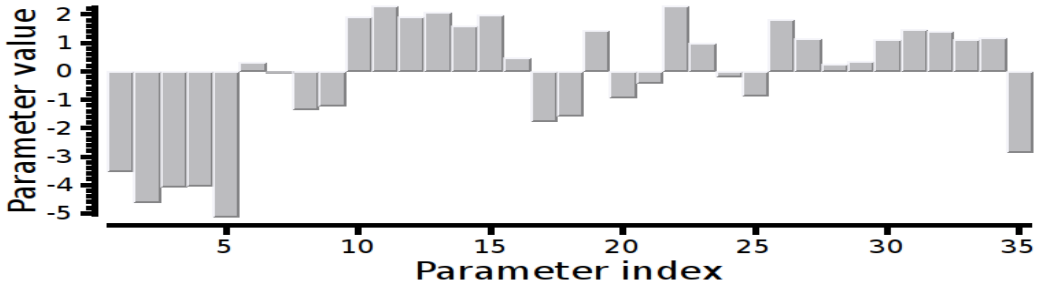
We discussed in the introductory chapter that GLM models provide a neurophysiological interpretation of their parameters. However, they consider neurons are independent. [Macke et al., 2011] added shared input to partly circumvent this inconvenience. The added dependencies comes then from the shared input. However, dependencies comes also from the connections between neurons. One way to improve the GLM models is to add Maximum Entropy parameters as statistical weight between neurons in order to take into account not only the stimulus, but also the network architecture (which includes dependencies between neurons).

Parameters/Stimulus relation

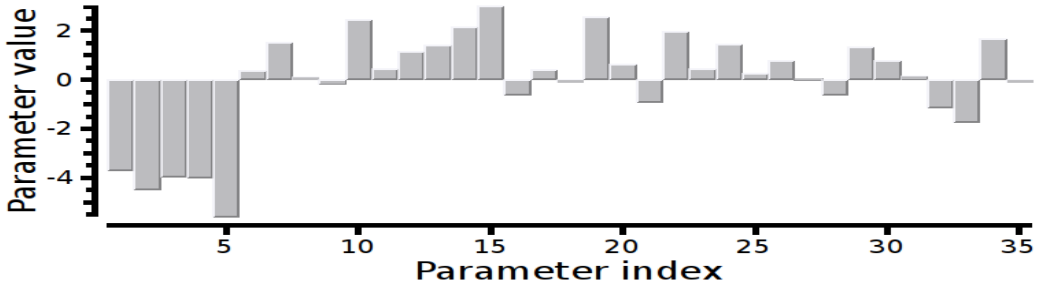
An important question to uncover is the relation between the stimulus and the Maximum Entropy parameters. From the data sets D_{11}, D_{12}, D_{13} and D_{14} which corresponds to 4 different stimuli on the same retina, if we fit a pairwise model with range $R = 2$ (for only 5 neurons), we can see that the parameters change from stimulus to another. This means that the probability distribution $P[R|S]$ represented by those coefficients could be investigated to infer the $P[S|R]$.



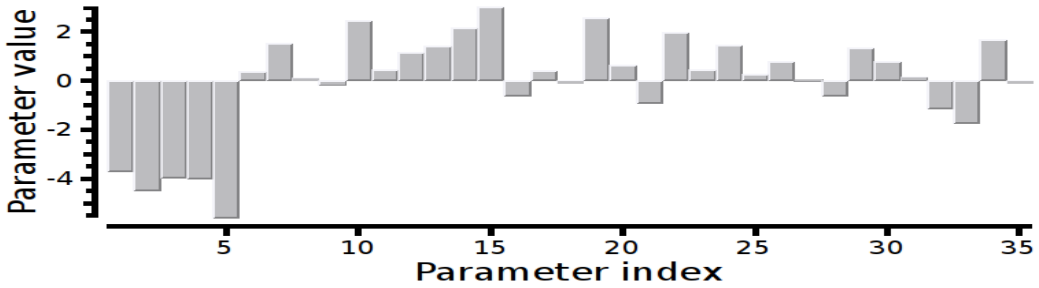
(a) Fit on D_{11}



(b) Fit on D_{12}



(c) Fit on D_{13}



(d) Fit on D_{14}

Figure 6.4: (a), (b), (c) and (d) shows respectively the parameters of a pairwise model of range 2 fitted with 4 different stimuli.

6.4 The future of *EnaS*

EnaS was born with the idea of providing retina researchers to analyze their data acquisition and understand the relation between stimulus and response in order to understand the retinal architecture. Researchers want to understand the retinal architecture because they want to create retinal implants and visual bio-inspired technologies.

As a co-developer of *EnaS*, aware of the increasing emergence of the visual neuroscience field and its future applications, I wanted to make from this software a starting point for a future potential project. This action will not be without my will and enthusiasm to perform a technology transfer action from research to industry. Figure 6.5 is an attestation that shows the prize I got from the "UNICE Foundation" and it was the highest prize in an "annual enterprise creation competition". One of the target application of the project was an *EnaS*-like project (whether adding new features to *EnaS* or developing another application in the spirit of spike train analysis and neural coding). In this section, I discuss an *EnaS* based application and the potential opportunity that could offer.

We can look at the opportunity that *EnaS* presents from two angles, implying two different business models:

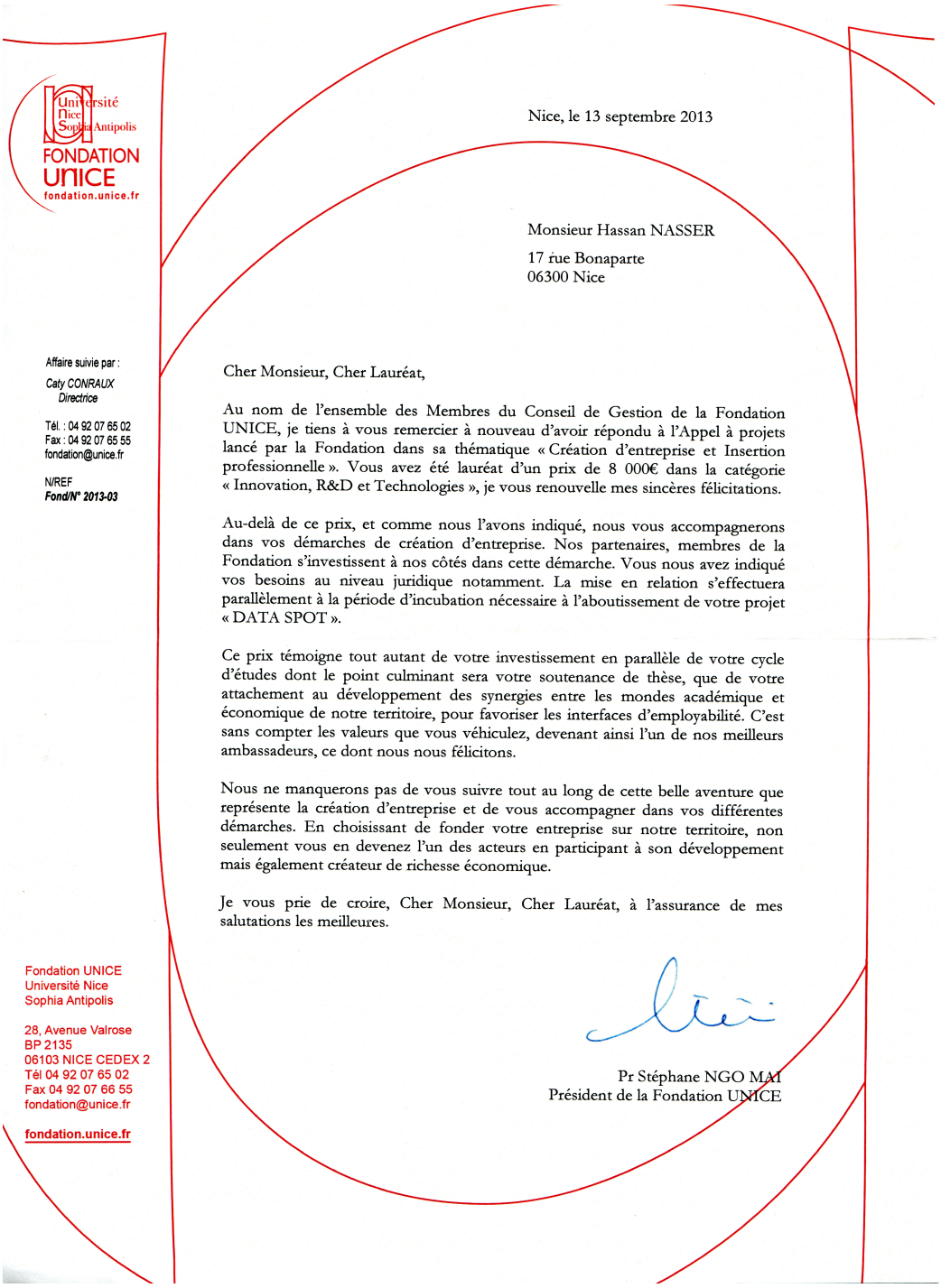
- Commercial: where the software could be taken in charge by a private company, develop it and commercialize it.

- Open-source: where we build a community around it by making it open source and adding continuously new advances in neural coding.

We can also imagine many configurations that mixes the two business models, e.g., a freemium[1] and community based software with the possibility of commercializing some modules. However, here we will discuss the first option (purely commercial). Before discussing it, it is very useful to present the currently available tools presented by direct and indirect competitors as well as target clients[2].

[1]Freemium is a pricing strategy by which a proprietary product or service (typically a digital offering such as software, media, games or web services) is provided free of charge, but money (premium) is charged for advanced functionalities or virtual goods. Source: `http://en.wikipedia.org`.

[2]Direct competitors are those who provide the same service. Indirect competitors are those who provide a similar service.



Université Nice Sophia Antipolis
FONDATION UNICE
fondation.unice.fr

Nice, le 13 septembre 2013

Monsieur Hassan NASSER
17 rue Bonaparte
06300 Nice

Affaire suivie par :
Caty CONRAUX
Directrice

Tél. : 04 92 07 65 02
Fax : 04 92 07 65 55
fondation@unice.fr

N/REF
Fond/N° 2013-03

Cher Monsieur, Cher Lauréat,

Au nom de l'ensemble des Membres du Conseil de Gestion de la Fondation UNICE, je tiens à vous remercier à nouveau d'avoir répondu à l'Appel à projets lancé par la Fondation dans sa thématique « Création d'entreprise et Insertion professionnelle ». Vous avez été lauréat d'un prix de 8 000€ dans la catégorie « Innovation, R&D et Technologies », je vous renouvelle mes sincères félicitations.

Au-delà de ce prix, et comme nous l'avons indiqué, nous vous accompagnerons dans vos démarches de création d'entreprise. Nos partenaires, membres de la Fondation s'investissent à nos côtés dans cette démarche. Vous nous avez indiqué vos besoins au niveau juridique notamment. La mise en relation s'effectuera parallèlement à la période d'incubation nécessaire à l'aboutissement de votre projet « DATA SPOT ».

Ce prix témoigne tout autant de votre investissement en parallèle de votre cycle d'études dont le point culminant sera votre soutenance de thèse, que de votre attachement au développement des synergies entre les mondes académique et économique de notre territoire, pour favoriser les interfaces d'employabilité. C'est sans compter les valeurs que vous véhiculez, devenant ainsi l'un de nos meilleurs ambassadeurs, ce dont nous nous félicitons.

Nous ne manquerons pas de vous suivre tout au long de cette belle aventure que représente la création d'entreprise et de vous accompagner dans vos différentes démarches. En choisissant de fonder votre entreprise sur notre territoire, non seulement vous en devenez l'un des acteurs en participant à son développement mais également créateur de richesse économique.

Je vous prie de croire, Cher Monsieur, Cher Lauréat, à l'assurance de mes salutations les meilleures.

Pr Stéphane NGO MAI
Président de la Fondation UNICE

Fondation UNICE
Université Nice
Sophia Antipolis

28, Avenue Valrose
BP 2135
06103 NICE CEDEX 2
Tél 04 92 07 65 02
Fax 04 92 07 66 55
fondation@unice.fr

fondation.unice.fr

Figure 6.5: The attestation of prize from the foundation UNICE.

6.4.1 Competitors and existing tools

Without distinguishing between what is commercial or community based, we show a (non-exhaustive) list of the main tools used in the market for analyzing spike trains (Table 6.1). The table 6.1 show that there are two type of competitors: those who are mainly hardware manufacturers and need to supply software to their clients so that they can acquire and analyze data and those who are only software providers and there is only one: Nex technologies. In addition, the NeuroExplorer is sold by many MEA manufacturers such as Multi Chanel systems and Plexon.

Company	# Electrodes	software
Alpha MED Scientific Inc.	64	**Mobius Qt** Spike sorting
Axion biosystems	64 and 768	**Axion integrated studio** Real time analysis module
3Brain	4096	**Brainwave** Real time analysis module
Multi Chanel systems	32, 60 120, 240	**NeurShare & NeuroExplorer** Data Analysis
Plexon	16 to 256	**Offline sorter** Spike sorting **NeuroExplorer** Data analysis
Bionic Technologies		NeuroShare
Cambridge Electronics Design		
Nex Technologies		**Empirical Data Analysis** NeuroExplorer
Bernstein Center for Computational Neuroscience		Spyke Viewer

Table 6.1: A table showing a (non-exhaustive) list of direct and indirect competitors. The table includes software and hardware manufacturers (because they also sell their own or their collaborators software).

6.4.2 The market and the product

The market of analyzing spike train is increasing over the year, especially with the increasing interest of research centers in exploiting MEA data. The

fact that the first application we used in *EnaS* is for retinal data, opens the opportunity not only for MEA analysis market, but also for other niche market such as: retinal prosthetic and vision based robotics.

EnaS in it current version offers a friendly and highly interactive environment for analyzing and fitting spike trains. Its added value is the work done by thesis, "Analyzing large scale with spatio-temporal constraints". This is surely not enough for a market entry, nor for a lean (go to market and improve based on feedback) development. Empirical statistics are offered by many suppliers that have been shortly presented in the table 6.1. If we found a way to serve researchers from using the results of fitting, then we answer the "so what" and we can have a concrete added value.

Such a product needs some more improvements in order to fit for clients needs. In fact, before studying the neural coding, the clients might be interested in the following applications:

- Spike sorting.
- Performing a wide range of empirical analysis and neural coding (such those provided by NeuroExplorer).
- Studying several methods of neural coding (time series coding, probabilistic modeling (for instance, the work done in this thesis)).
- Integrating a retinal simulator (*VirtualRetina* for instance).
- Possibility of using the computed parameters in the retinal simulator.
- Data adapter that offers a interface to read/write the most common data types in neuroscience community.

Taking into account the needs cited up, we can imagine a future product that satisfies the clients needs after data acquisitions. In fact, one of the problem that the analysts often encounter is when they use many software (one for spike sorting, one for empirical analysis ...) is the conflicts between formats. A full package software not only fulfill the A-Z needs but circumvent the problem of formats.

6.5 General conclusion

In this thesis, we were able to contribute at several levels:

- A mathematical framework, that allows to compute large scale target distribution using Montecarlo algorithm.

- The parallelization of the Montecarlo algorithm on multi-processors computers and clusters.
- A method to fit parameters for large scale Maximum entropy models.
- Method for evaluating the goodness of fit such as Kullback-Leibler divergence, for large networks.
- Contribution to the development of *EnaS* software, a highly interactive and friendly tool to allow analyzing large scale data.
- Results of analysis applied on large scale data set from experiments.

The main asset of this thesis is that we were able to provide a tool to allow analyzing large scale spike trains with Maximum Entropy. The value that this method adds is its capacity to determine the dominant interactions between neurons taking into account memory effects. This opens the opportunity to go further at the level of many projects, which we list here:

- Working on the method itself:
  - Optimize the Montecarlo algorithm. Although we did our best to provide an optimized Montecarlo algorithm, but we believe we still can optimize more.
  - Rethinking of the model shape: we have shown that the canonical form potentials cause some troubles for the parameters estimation. We believe that a filtering method should be developed, not using filtering, but using other framework such that iterative features selection.
  - Study the effect of binning: together with the filtering, binning should be studied rigorously and it might affect also the features selection method since, as we have shown, empirical data contains less monomials when we take binning equal to 1.
- Finding a way to use the parameters: the parameters represent the statistical weight that a monomial have in the data. Those parameters might be used to be integrated in the GLM model or as a statistical connectivities in the VirtualRetina.
- Improve the software. *EnaS* would be better if we add more features such as time-series analysis, other neural coding schemes taken from literature, spike sorting, retina modeling plugins (such as VirtualRetina), possibility of identifying cell types, stimulus design adds .... In this

way, it would be a one place (and maybe a reference place) to study the retina.

We hope that we were clear in explaining our ideas and perspectives and we wish that this work is useful and inspiring for the fellow projects.

6.6 The list of publications

Below is the list of official publications in international journals, international and national conferences.

6.6.1 Papers in international journals

- Nasser H, Cessac B and Kraria S. EnaS: a new software for neural population analysis in large scale spiking networks. Hassan Nasser, Selim Kraria and Bruno Cessac. In preparation.

- Nasser H, Cessac B. Parameter Estimation for Spatio-Temporal Maximum Entropy Distributions: Application to Neural Spike Trains. Entropy. 2014; 16(4):2244-2277.

- Nasser H, Cessac B and Marre O. Spatio-temporal spike train analysis for large scale networks using the maximum entropy principle and Monte Carlo method. Journal of Statistical Mechanics: Theory and Experiment. 2013; 03006.

6.6.2 Conference papers/posters

- Toward a realistic input for visual cortex models, Hassan Nasser, Bruno Cessac, Bodgan Kolomiets, Pierre Kornprobst, Serge Picaud, TAUC 2010.

- Spike trains statistics in Integrate and Fire Models: exact results, Bruno Cessac, Hassan Nasser, Juan-Carlos Vasquez, Proceedings of the NeuroComp2010 Conference.

- Parametric estimation of Spike train statistics by Gibbs distributions: an application to bio-inspired and experimental data, Juan-Carlos Vasquez, Hassan Nasser, Adrian Palacios, Bruno Cessac, Thierry Viéville, Horacio Rostro-Gonzalez, Proceedings of Neurocomp 2010 (Lyon).

- Analyzing large-scale spike trains data with spatio-temporal constraints, Hassan Nasser, Olivier Marre, Bruno Cessac, SCNE 2012. Wien, Austria, September 2012. Sensory Coding and Natural Environment.

- Spatio temporal Gibbs distribution analysis of spike trains using Monte Carlo method, Hassan Nasser, Olivier Marre, Michael J. Berry II, Bruno Cessac, AREADNE 2012 Research in Encoding And Decoding of Neural Ensembles.

- Spatio-Temporal modeling of large-scale retinal networks using Monte Carlo principle, Hassan Nasser, Olivier Marre, Bruno Cessac, Inauguration INT, 2012.

- Analyzing large-scale spike trains data with spatio-temporal constraints, Hassan Nasser, Olivier Marre, Bruno Cessac, NEUROCOMP 2012. Bordeaux, France, October, 2012. The NeuroComp/KEOpS 12 workshop.

- On the ubiquity of Gibbs distributions in spike train statistics, Bruno Cessac, Rodrigo Cofré, Hassan Nasser, 3rd annual meeting of the GDR 2904 "Multi-electrodes systems and signal processing to study neural networks", 2012, Marseille.

Bibliography

[Ackley et al., 1985] Ackley, H., Hinton, E., and Sejnowski, J. (1985). A learning algorithm for boltzmann machines. Cognitive Science, 9(1):147–169.

[Ackley et al., 1987] Ackley, H., Hinton, E., and Sejnowski, J. (1987). a mean field theory learning algorithm for neural network. Complex systems, 1(5):995–1019.

[Ahmadian et al., 2011] Ahmadian, Y., Pillow, J. W., and Paninski, L. (2011). Efficient Markov Chain Monte Carlo Methods for Decoding Neural Spike Trains. Neural Computation, 23(1):46–96.

[Aldworth et al., 2011] Aldworth, Z. N., Dimitrov, A. G., Cummins, G. I., Gedeon, T., and Miller, J. P. (2011). Temporal encoding in a nervous system. PLoS computational biology, 7(5):e1002041.

[Amari, 2001] Amari, S. (2001). Information geometry on hierarchy of probability distributions. IEEE Transactions on Information Theory, 47(5):1701–1711.

[Beck and Schloegl, 1995] Beck, C. and Schloegl, F. (1995). Thermodynamics of Chaotic Systems: An Introduction. Cambridge University Press, Cambridge.

[Berger et al., 1996] Berger, A. L., Pietra, S. A. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. Computational lainguistics, 22:39–71.

[Bloomfield and Völgyi, 2009] Bloomfield, S. A. and Völgyi, B. (2009). The diverse functional roles and regulation of neuronal gap junctions in the retina. Nature Reviews Neuroscience, 10(7):495–506.

[Bowen, 1975] Bowen, R. (1975). Equilibrium states and the ergodic theory of Anosov diffeomorphisms, volume 470 of Lect. Notes.in Math. Springer-Verlag, New York.

[Bowen, 2008] Bowen, R. (2008). Equilibrium states and the ergodic theory of Anosov diffeomorphisms. Second revised version., volume 470 of Lect. Notes.in Math. Springer-Verlag.

[Brillinger, 1988] Brillinger, D. R. (1988). Maximum likelihood analysis of spike trains of interacting nerve cells. Biol Cybern, 59(3):189–200.

[Brillinger, 1992] Brillinger, D. R. (1992). Nerve Cell Spike Train Data Analysis - a Progression of Technique. J Amer Statist Assn, 87(418):260–271.

[Broderick et al., 2007] Broderick, T., Dudik, M., Tkacik, G., Schapire, R. E., and Bialek, W. (2007). Faster solutions of the inverse pairwise ising problem. arXiv preprint arXiv:0712.2437.

[Brown et al., 2003] Brown, E. N., Barbieri, R., Eden, U. T., and Frank, L. M. (2003). Likelihood methods for neural spike train data analysis. Computational neuroscience: A comprehensive approach, pages 253–286.

[Cessac and Cofre, 2013] Cessac, B. and Cofre, R. (2013). Estimating maximum entropy distributions from periodic orbits in spike trains. research report RR-8329, INRIA.

[Chazottes and Keller, 2008] Chazottes, J. and Keller, G. (2008). Pressure and equilibrium states in ergodic theory. Israel Journal of Mathematics, 131(1).

[Chen and Rosenfeld, 1999] Chen, S. F. and Rosenfeld, R. (1999). Efficient sampling and feature selection in whole sentence maximum entropy language models.

[Chichilnisky, 2001] Chichilnisky, E. J. (2001). A simple white noise analysis of neuronal light responses. Network: Comput. Neural Syst., 12:199–213.

[Cocco et al., 2009] Cocco, S., Leibler, S., and Monasson, R. (2009). Neuronal couplings between retinal ganglion cells inferred by efficient inverse statistical physics methods. PNAS, 106(33):14058–14062.

[Cofré and Cessac, 2013] Cofré, R. and Cessac, B. (2013). Dynamics and spike trains statistics in conductance-based integrate-and-fire neural networks with chemical and electric synapses. Chaos, Solitons and Fractals, 50(8):13–31.

[Collins et al., 2002] Collins, M., Schapire, R. E., and Singer, Y. (2002). Logistic Regression, AdaBoost and Bregman Distances. Machine Learning, 48:253–285.

[Cornfeld et al., 1982] Cornfeld, I. P., Fomin, S. V., and Sinai, Y. G. (1982). Ergodic Theory. Springer, Berlin, Heidelberg, New York.

[Croner et al., 1993] Croner, L., Purpura, K., and Kaplan, E. (1993). Response variability in retinal ganglion cells of primates. Proceedings of the National Academy of Sciences of the United States of America, 90(17):8128–8130.

[Csiszár and Talata, 2006] Csiszár, I. and Talata, Z. (2006). Context tree estimation for not necessarily finite memory processes, via bic and mdl. Information Theory, IEEE Transactions on, 52(3):1007–1016.

[Dudík et al., 2004] Dudík, M., Phillips, S., and Schapire, R. (2004). Performance guarantees for regularized maximum entropy density estimation. In Proceedings of the 17th Annual Conference on Computational Learning Theory.

[Fairhall et al., 2006] Fairhall, A. L., Burlingame, A. C., Narasimhan, R., Harris, R. A., Puchalla, J. L., and Berry, M. J. (2006). Selectivity for Multiple Stimulus Features in Retinal Ganglion Cells. J Neurophysiol, 96(5):2724–2738.

[Ferrea et al., 2012] Ferrea, E., Maccione, A., Medrihan, L., Nieus, T., Ghezzi, D., Baldelli, P., Benfenati, F., and Berdondini, L. (2012). Large-scale, high-resolution electrophysiological imaging of field potentials in brain slices with microelectronic multielectrode arrays. Frontiers in Neural Circuits., 6(80).

[Galves et al., 2012] Galves, A., Galves, C., García, J. E., Garcia, N. L., and Leonardi, F. (2012). Context tree selection and linguistic rhythm retrieval from written texts. The Annals of Applied Statistics, 6(1):186–209.

[Ganmor et al., 2011a] Ganmor, E., Segev, R., and Schneidman, E. (2011a). The architecture of functional interaction networks in the retina. The journal of neuroscience, 31(8):3044–3054.

[Ganmor et al., 2011b] Ganmor, E., Segev, R., and Schneidman, E. (2011b). Sparse low-order interaction network underlies a highly correlated and learnable neural population code. PNAS, 108(23):9679–9684.

[Gantmacher, 1998] Gantmacher, F. R. (1998). the theory of matrices. AMS Chelsea Publishing, Providence, RI.

[Garibaldi and Penco, 1985] Garibaldi, U. and Penco, M. A. (1985). Probability theory and physics between bernoulli and laplace: The contribution of j. h. lambert (1728-1777). In Proc. Fifth National Congress on the History of Physics, volume 9, pages 341–346.

[Georgii, 1988] Georgii, H.-O. (1988). Gibbs measures and phase transitions. De Gruyter Studies in Mathematics:9. Berlin; New York.

[Gollisch and Meister, 2008] Gollisch, T. and Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. Science, 319(5866):1108–1111.

[Hastings, 1970] Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. Biometrika, 57:97–109.

[Higuchi and Mezard, 2009] Higuchi, S. and Mezard, M. (2009). Susceptibility propagation for constraint satisfaction problems. CoRR, pages –1–1.

[Jaynes, 1957] Jaynes, E. (1957). Information theory and statistical mechanics. Phys. Rev., 106:620.

[Kappen and Rodriguez, 1998] Kappen, H. and Rodriguez, F. (1998). Boltzmann machine learning using mean field theory and linear response correction. In Advances in Neural Information Processing Systems, volume 12, pages 280–286. The MIT Press.

[Keller, 1998] Keller, G. (1998). Equilibrium States in Ergodic Theory. Cambridge University Press.

[Koeling, 2000] Koeling, R. (2000). Chunking with maximum entropy models.

[Macke et al., 2011] Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., and Sahani, M. (2011). Empirical models of spiking in neural populations. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, Advances in Neural Information Processing Systems 24, pages 1350–1358.

[Marre et al., 2009] Marre, O., El Boustani, S., Frégnac, Y., and Destexhe, A. (2009). Prediction of spatiotemporal patterns of neural activity from pairwise correlations. Physical review letters, 102(13).

[Mayer and Urbański, 2010] Mayer, V. and Urbański, M. (2010). Thermodynamical formalism and multifractal analysis for meromorphic functions of finite order. Memoirs of the American Mathematical Society, 203(954).

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. A. (1989). Generalized linear models (Second edition). London: Chapman & Hall.

[Mézard and Mora, 2009] Mézard, M. and Mora, T. (2009). Constraint satisfaction problems and neural networks: a statistical physics perspective. J. Physiol. Paris, 103:107–113.

[Multi Channel Systems, 2013] Multi Channel Systems, M. (2013). Mea application note retina recordings (micro electroretinograms) from rattus norvegicus. Technical report, Multi Channel Systems MCS GmbH.

[Nakahara and Amari, 2001] Nakahara, H. and Amari, S. (2001). Information-geometric decomposition in spike analysis. In NIPS, pages 253–260.

[Nasser et al., 2013] Nasser, H., Marre, O., and Cessac, B. (2013). Spatio-temporal spike train analysis for large scale networks using the maximum entropy principle and montecarlo method. Journal of Statistical Mechanics: Theory and Experiment, 2013(03):P03006.

[Nirenberg and Victor, 2007] Nirenberg, S. H. and Victor, J. D. (2007). Analyzing the activity of large populations of neurons: how tractable is the problem? Current Opinion in Neurobiology, 17(4):397–400.

[Ohiorhenuan et al., 2010] Ohiorhenuan, I. E., Mechler, F., Purpura, K. P., Schmid, A. M., Hu, Q., and Victor, J. D. (2010). Sparse coding and high-order correlations in fine-scale cortical networks. Nature, 466(7):617–621.

[Paninski, 2004] Paninski, L. (2004). Maximum likelihood estimation of cascade point-process neural encoding models. Network: Comput. Neural Syst., 15(04):243–262.

[Paninski et al., 2004] Paninski, L., Fellows, M., Shoham, S., Hatsopoulos, N., and Donoghue, J. (2004). Superlinear population encoding of dynamic hand trajectory in primary motor cortex. J. Neurosci., 24:8551–8561.

[Panzeri et al., 2010] Panzeri, S., Brunel, N., Logothetis, N. K., and Kayser, C. (2010). Sensory neural codes using multiplexed temporal scales. Trends in neurosciences, 33(3):111–120.

[Parry and Pollicott, 1990] Parry, W. and Pollicott, M. (1990). Zeta functions and the periodic orbit structure of hyperbolic dynamics, volume 187–188. Asterisque.

[Pillow et al., 2005] Pillow, J., Paninski, L., Uzzell, V., Simoncelli, E., and Chichilnisky, E. (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. J. Neurosci, 25:11003–11013.

[Pillow et al., 2011] Pillow, J. W., Ahmadian, Y., and Paninski, L. (2011). Model-based decoding, information estimation, and change-point detection techniques for multineuron spike trains. Neural Comput., 23(1):1–45.

[Pillow et al., 2008] Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E. J., and Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signaling in a complete neuronal population. Nature, 454(7206):995–999.

[Pressé et al., 2013] Pressé, S., Ghosh, K., Lee, J., and Dill, K. A. (2013). Principles of maximum entropy and maximum caliber in statistical physics. Reviews of Modern Physics, 85(3):1115.

[Puchalla et al., 2005] Puchalla, J. L., Schneidman, E., Harris, R. A., and Berry, M. J. (2005). Redundancy in the population code of the retina. Neuron, 46(3):493–504.

[Quiroga et al., 2004] Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural computation, 16(8):1661–1687.

[Rosenfeld et al., 1994] Rosenfeld, R., Carbonell, J., and Rudnicky, A. (1994). Adaptive statistical language modeling: A maximum entropy approach. Technical report, School of Computer Science, Carnegie Mellon University.

[Roudi and Hertz, 2011] Roudi, Y. and Hertz, J. (2011). Mean field theory for non-equilibrium network reconstruction. Phys. Rev. Lett., 106(048702).

[Roudi et al., 2009a] Roudi, Y., Nirenberg, S., and Latham, P. (2009a). Pairwise maximum entropy models for studying large biological systems: when they can work and when they can't. PLOS Computational Biology, 5(5).

[Roudi et al., 2009b] Roudi, Y., Tyrcha, J., and Hertz, J. A. (2009b). Ising model for neural data: Model quality and approximate methods for extracting functional connectivity. Physical Review E, page 051915.

[Ruelle, 1978] Ruelle, D. (1978). Thermodynamic formalism. Addison-Wesley,Reading, Massachusetts.

[Schneidman et al., 2006] Schneidman, E., Berry, M., Segev, R., and Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. Nature, 440(7087):1007–1012.

[Seneta, 2006] Seneta, E. (2006). Non-negative Matrices and Markov Chains. Springer.

[Sessak and Monasson, 2009] Sessak, V. and Monasson, R. (2009). Small-correlation expansions for the inverse ising problem. Journal of Physics A: Mathematical and Theoretical, 42(5):055001.

[Sherrington and Kirkpatrick, 1975] Sherrington, D. and Kirkpatrick, S. (1975). Solvable model of a spin-glass. Physical Review Letters, 35(26):1792+.

[Shlens et al., 2006] Shlens, J., Field, G., Gauthier, J., Grivich, M., Petrusca, D., Sher, A., Litke, A., and Chichilnisky, E. (2006). The structure of multi-neuron firing patterns in primate retina. Journal of Neuroscience, 26(32):8254.

[Simoncelli et al., 2004] Simoncelli, E. P., Paninski, J. P., Pillow, J., and Schwartz, O. (2004). Characterization of Neural Responses with Stochastic Stimuli. The cognitive neurosciences.

[Stevenson and Kording, 2011] Stevenson, I. H. and Kording, K. P. (2011). How advances in neural recording affect data analysis. Nature Neuroscience, 14(2):139–142.

[Strong et al., 1998] Strong, S., Koberle, R., de Ruyter van Steveninck, R., and Bialek, W. (1998). Entropy and information in neural spike trains. Phys. Rev. Let, 80(1):197–200.

[Tanaka, 1998] Tanaka, T. (1998). A theory of mean field approximation. In Advances in Neural Information Processing Systems 11, volume 12, pages 351–357. The MIT press.

[Tang et al., 2008] Tang, A., Jackson, D., Hobbs, J., Chen, W., Smith, J. L., Patel, H., Prieto, A., Petrusca, D., Grivich, M. I., Sher, A., Hottowy, P., Dabrowski, W., Litke, A. M., and Beggs, J. M. (2008). A maximum entropy model applied to spatial and temporal correlations from cortical networks *In Vitro*. The Journal of Neuroscience, 28(2):505–518.

[Theunissen et al., 2001] Theunissen, F. E., David, S. V., Singh, N. C., Hsu, A., Vinje, W. E., and Gallant, J. L. (2001). Estimating spatio-temporal

receptive fields of auditory and visual neurons from their responses to natural stimuli. Network, 12(3):289–316.

[Thouless et al., 1977] Thouless, D. J., Anderson, P. W., and Palmer, R. G. (1977). Solution of solvable model of a spin glass. Phil. Mag., 35:593–601.

[Tkacik et al., 2010] Tkacik, G., Prentice, J. S., Balasubramanian, V., and Schneidman, E. (2010). Optimal population coding by noisy spiking neurons. PNAS, 107(32):14419–14424.

[Tkačik et al., 2009] Tkačik, G., Schneidman, E., Berry II, M. J., and Bialek, W. (2009). Spin glass models for a network of real neurons. arXiv preprint arXiv:0912.5409.

[Truccolo et al., 2005] Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., and Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects. J Neurophysiol, 93:1074–1089.

[Vasquez et al., 2012] Vasquez, J. C., Marre, O., Palacios, A. G., Berry, M. J., and Cessac, B. (2012). Gibbs distribution analysis of temporal correlation structure on multicell spike trains from retina ganglion cells. J. Physiol. Paris, 106(3-4):120–127.

[Vasquez et al., 2010] Vasquez, J.-C., Viéville, T., and Cessac, B. (2010). Entropy-based parametric estimation of spike train statistics. INRIA Research Report.

[Welling and Teh, 2003] Welling, M. and Teh, Y. W. (2003). Approximate inference in boltzmann machines. Artificial Intelligence, 143(1):19 – 50.

[Yu et al., 2008] Yu, S., Huang, D., Singer, W., and Nikolic, D. (2008). A small world of neuronal synchrony. Cereb. Cortex.

[Zhou and Wu, 2003] Zhou, Y. and Wu, L. (2003). A fast algorithm for feature selection in conditional maximum entropy modeling. In in Proceedings of the EMNLP 2003, pages 153–159.