Network of interacting neurons with random synaptic weights

Grasziechi Paolo & Leocata Marta & Mascart Cyrille
Supervisors: Chevalier Julien & Delarue Fran cois & Tanre Etienne

Abstract

Derived from the work of Hodgkin andHuxley, several examples of the fokker-planck equation can be used to model the evolution in time of the potential of the membrane of a neuron. The solutions to the Fokker-Planck equation describing the behaviour of the potential of the membrane of a neuron can sometimes degenerate and cease to exist after a finite time. We focus out study on the conditions for such ablow-up to exist in a system driven by a classical mean-field equation modeling the behaviour of a network of integrate and fire neurons. After a brief introduction to the model we begin by an analytical study of the equation in order to restrict the research for these conditions. We then step on these results to develop an algorithm for simulating a large network of integrate and fire neurons driven by this equation before showing the results of the runs of this simulator.

Contents

List of Figures

List of Tables

Introduction

One of the first models for neurons was introduced by Louis Lapicque in 1907 and called Integrate and Fire. Neurons are represented in time by the simple electrical equation

$$I(t) = C_m \frac{dX_m}{dt}$$

which is just the time derivative of the law of capacitance. A positive current being applied to the membrane, it's potential is going to increase until it reaches a certain threshold value X_T, at which point a dirac delta function happens and the voltage of the mambrane is reset to a resting potential. This model is the basis of a large variety of other neural network models invented to model more precisely certain behaviours of neurons and neural networks, as memory, leaking, etc.
We describe the discrete system of neurons by keeping track of membrane potentials and of the way they evolve in time: in this sense, neuron i (for $i = 1, ..., N$) is described by the process X^i which actually stands for the membrane potential of neuron i itself. The dynamics of these processes is the following:

$$dX^i(t) = b(X^i(t))dt + \sum_j \sum_k \beta \frac{\alpha^{i,j}}{N} \delta_0(t - \tau_k^j)dt + \sigma dW_t^i \quad i = 1, \cdots, N \quad t \geq 0$$

with initial condition $X(0) = X_0 < 1$; here $b(x) = c - \lambda x$, $\lambda > 0$ is a constant, $\alpha^{i,j}$ represents the (random) synaptic weight between neurons i and j, while β is a constant to calibrate the weight of the connections. The idea is to study the behavior of the system considering different kind of randomness on $\alpha^{i,j}$. We will focus on the following cases:

1. $\alpha^{i,j} \sim \mathcal{B}(p)$ i.i.d;
2. $\alpha^{i,j}$ dependent: namely $\alpha^{i,j} = \alpha^i \alpha^j$;
3. $\alpha^{i,j} = p\alpha^i$ with $\alpha^i \sim \mathcal{B}(p)$;
4. $\alpha^{i,j} \sim \mathcal{B}(p_N)$.

Chapter 1

Model

In this study we are interrested in a large (possibly infinite) network of interacting integrate and fire neurons. [**?**] and [**?**] proposed an equation describing the evolution in time of the potential X_i of the i^{th} neuron in a network of N

$$\begin{cases} \frac{d}{dt}X_i(t) = -\lambda X_i(t) + \frac{\alpha}{N}\sum_j\sum_i \delta_0(t-\tau_k^j) + \frac{\beta}{N}\sum_{j\neq i} X_j(t) + I_i^{ext}(t) + \sigma\mu_i(t) & \text{if } X_i < X_T \\ X_i(t^+) = X_R & \text{if } X_i \text{ reaches } X_T \text{ at time t} \end{cases} \tag{1.1}$$

[**?**] and [**?**] give more detail about the physical signification of the terms in the equation. As an overview, $X_i(t)$ is the function associated with the evolution in time of the potential of the membrane of the neuron i, $I_i^{ext}(t) + \sigma\mu_i(t)$ is the effect of electrical currents outside of the studied network (mean value + gaussian white noise), $\frac{\alpha}{N}\sum_j\sum_i \delta_0(t-\tau_k^j)$

1.1 Motivations

Here we are interested only on the spiking behaviour of the system, and so the mean field equation considered thereafter in this document is a simplified form of the one presented just above, as some terms are irreleveant to the question. In this paper we are going to focus on what sets of parameters favors the apparition of a blowing up of the system. The influence of the interaction term is a big part of the study, but the influence of other parameters, such as the b function, the noise, and the topology of the network are also looked at.
In the deterministic case, the blow up is quite easily defined by the limit to a time value t_b of the variations in the spike rate equals to infinity, in other words

$$\lim_{t\to t_b} \frac{de}{dt} = \infty \tag{1.2}$$

This behaviour of the PDE has been described in [**?**], while others have been interrested in this exact phenomenon from a PDE perspective. Indeed, systems ruled by this kind of equations do not automatically blow-up; actually some conditions on the parameters must be met in order to observe this phenomenon.

1.2 Mean field equation

That being said, we can now pose the real equation under study in this report.

$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \sigma W_t^i + \sum_{j=1}^N J^{j\to i}M_t^j - M_t^i(X_T^i - X_R^i)$$

Here b is Lipschitz continuous, and $\forall X, b(X) = -\lambda(X-a), (\lambda, a) \in \mathbb{R}_+$. This function will make the potential drift towards the value a. The J term models the interactions between neurons, their values depends on the number of neurons in the system and which neurons are actually involved in the interaction.
Throughout the event several values of interaction have been tested. They generally are Bernoullis variables, determined at the creation of the system.

Chapter 2

Mathematical inquiries

2.1 Independent Random weight: $\alpha^{i,j} \sim \mathcal{B}(p)$

Let us consider the case where connections are i.i.d $\sim \mathcal{B}(p)$. The first purpose is to conjecture what is the limit equation for the network. By some heuristic calculation, we expect to see the term:

$$p \cdot \beta \cdot \mathbb{E}[M_t]$$

because for $N \to \infty$, the neurons become asymptotically independent; this means that, applying a law of large numbers,

$$\frac{1}{N}\mathbb{E}\left[\int_0^t \sum_j \sum_k \beta\alpha^{i,j}\delta(s-\tau_k^j)ds\right] = \frac{1}{N}\mathbb{E}\left[\sum_j \sum_k \beta\alpha^{i,j}\mathbb{I}_{\tau_k^j \le t}\right] = p \cdot \beta \cdot \mathbb{E}[M_t].$$

So, as a limit equation for the network, we expect the following:

$$X_t = X_0 + \int_0^t b(X_s)ds + \beta \cdot p \cdot \mathbb{E}[M_t] + \sigma W_t - M_t$$

with $M_t = \sum_{k\le 1} \mathbb{I}_{[0,t]}(\tau_k)$. To justify more rigorously the conjectured equation, we propose the following argument based on a simple model for propagation of chaos.

2.1.1 Derivation of the limit equation in the simpler model

Propagation of chaos is the phenomenon where some interacting particles become independent at the limit for the number of particles going to infinity.
To get an idea of the behaviour of particles in neuronal networks, we consider the simpler model

$$dX_t^i = \frac{1}{N}\sum_{i=1}^N \alpha^{i,j} X_t^j \, dt + dW_t^i, \tag{2.1}$$

where $(\alpha^{i,j})_{i,j=1,\dots,N}$ is a family of i.i.d. random variables with finite first momentum and $(W^i)_i$ is a family of independent standard Brownian motions. The family of the r.v.(s) $\alpha^{i,j}$ and that of the Brownian motions W^i are assumed to be independent.

We use the technique of **coupling**. That is, we want to show convergence of (2.1) to

$$d\bar{X}_t = \mathbb{E}[\alpha]\mathbb{E}[\bar{X}_t]dt + dW_t, \tag{2.2}$$

where α is another r.v. independent of all the $\alpha^{i,j}$ and of Brownian motions and where W is another Brownian motion, with the same independence assumptions. To prove the convergence stated above, let's also introduce the SDE

$$d\bar{X}_t^i = \mathbb{E}[\alpha]\mathbb{E}[\bar{X}_t^i]dt + dW_t^i \tag{2.3}$$

and let's follow the strategy consisting in proving some kind of "closeness" of (2.2) with (2.3) and of (2.3) with (2.1). That's what we mean by coupling.
Our "closeness" concept will be "in law" and, at this point, we can already state that $\bar{X}$ e $\bar{X}^i$ have the same

law: we therefore just need to consider X^i and $\bar{X}^i$.

We now observe that all the processes $\bar{X}$ and $(\bar{X}^i)_i$ are independent of all the $(\alpha^{i,j})_{i,j}$.
We also know that

$$\begin{aligned} d\left(X_t^i - \bar{X}_t^i\right) &= \left(\frac{1}{N}\sum_{j=1}^N \alpha^{i,j} X_t^j - \mathbb{E}\left[\alpha \bar{X}_t^i\right]\right) dt \\ &= \left[\left(\frac{1}{N}\sum_{j=1}^N \alpha^{i,j} X_t^j - \frac{1}{N}\sum_{j=1}^N \alpha^{i,j} \bar{X}_t^j\right) + \left(\frac{1}{N}\sum_{j=1}^N \alpha^{i,j} \bar{X}_t^j - \mathbb{E}\left[\alpha \bar{X}_t^i\right]\right)\right] dt. \end{aligned}$$

If $\alpha^{i,j}$ are bounded by a constant C (or if $\frac{1}{N}\sum_{j=1}^N |\alpha^{i,j}| \leq C$, where C is a random variable not depending on i nor on N), then

$$\left|X_t^i - \bar{X}_t^i\right| \leq \int_0^t \frac{C}{N}\sum_{j=1}^N \left|X_s^j - \bar{X}_s^j\right| ds + \int_0^t \left|\frac{1}{N}\sum_{j=1}^N \alpha^{i,j} \bar{X}_s^j - \mathbb{E}\left[\alpha \bar{X}_s^i\right]\right| ds.$$

By summing over i and dividing by N, also defining δ_t to be the function $\frac{1}{N}\sum_i \left|X_t^i - \bar{X}_t^i\right|$, we get

$$\delta_t \leq C\int_0^t \delta_s ds + \frac{1}{N^2}\int_0^t \sum_{i=1}^N \left|\sum_{j=1}^N \left(\alpha^{i,j}\bar{X}_s^j - \mathbb{E}\left[\alpha \bar{X}_s^i\right]\right)\right| ds.$$

By using Gronwall's lemma, we then deduce that

$$\delta_t \leq \frac{\exp(Ct)}{N^2}\int_0^t \sum_{i=1}^N \left|\sum_{j=1}^N \left(\alpha^{i,j}\bar{X}_s^j - \mathbb{E}\left[\alpha \bar{X}_s^i\right]\right)\right| ds.$$

Taking the expectation and using the Cauchy-Schwarz inequality, we have the following:

$$\mathbb{E}[\delta_t] \leq \frac{\left(\mathbb{E}\left[\exp(2Ct)\right]\right)^{1/2}}{N}\int_0^t \sum_{i=1}^N \left(\frac{1}{N^2}\mathbb{E}\left[\left(\sum_{j=1}^N \left(\alpha^{i,j}\bar{X}_s^j - \mathbb{E}\left[\alpha \bar{X}_s^i\right]\right)\right)^2\right]\right)^{1/2} ds.$$

It is therefore useful to investigate the second moment of $\left(\alpha^{i,j}\bar{X}_s^j - \mathbb{E}\left[\alpha \bar{X}_s^i\right]\right)$, which is a finite value C_2 **(EXERCISE)**. Hence:

1

$$\mathbb{E}[\delta_t] \leq \frac{\left(\mathbb{E}\left[\exp(2Ct)\right]\right)^{1/2}}{N}\int_0^t \sum_{i=1}^N \left(\frac{1}{N^2} 4NC_2\right)^{1/2} ds \leq \frac{\left(\mathbb{E}\left[\exp(2Ct)\right]\right)^{1/2} 2C_2^{1/2} t}{N^{1/2}}.$$

NB It holds that

$$\delta_t \geq W_1\left(\frac{1}{N}\sum_{i=1}^N \delta_{X_t^i}, \frac{1}{N}\sum_{i=1}^N \delta_{\bar{X}_t^i}\right),$$

where $W_1(\cdot,\cdot)$ is the 1-Wasserstein distance. We have therefore proved that the Wasserstein distance goes to 0, since δ_t does.

2.1.2 Blow-up Argument

We will here present a probabilistic argument for the blow-up, based on the proof presented by Carrilo, Perthame et al. in [**?**]

verificare bibliografi

Theorem 1. *Assume that drift coefficient satisfies*

$$b(v) \geq -\lambda v, \quad \textit{for all } -\infty < v \leq 1.$$

If the initial condition is concentrated around the threshold 1*, there is no global in time solution of the SDE*

$$dX_t = b(X_t)dt + \beta \cdot p \cdot e'(t)dt + dW_t - dM_t, \qquad t \geq 0,$$

with a deterministic initial condition $X(0) = x_0 < 1$*, with* $\beta \in \mathbb{R}^+$.

Proof. The idea is to retrace the proof of Carrillo, Perthame et al.
The object of our study is $\mathbb{E}[\varphi_\mu(X_t)]$ with $\varphi_\mu(x) = \exp(\mu x)$.
Remember that

$$X_t = X_0 + \int_0^t \Big(b(X_s) + \beta \cdot pe'(s)\Big)ds + W_t - M_t$$

with $M_t = \int_0^t \int_{\mathbb{R}_+} g(X_{s-}, z)N(ds, dz)$. Now we apply the Ito-formula to $\varphi_\mu(X_t)$ (for simplicity, I will omitt μ in the computation)

$$\varphi(X_t) = \varphi(X_0) + \int_0^t \underbrace{\varphi'(X_s)}_{\mu\varphi(X_s)} \Big(b(X_s) + \alpha \cdot pe'(s)\Big)ds + \int_0^t \underbrace{\varphi'(X_s)}_{\mu\varphi(X_s)} dW_s + \frac{1}{2}\int_0^t \underbrace{\varphi''(X_s)}_{\mu^2\varphi(X_s)} ds +$$
$$\int_0^t \int_{\mathbb{R}_+} \underbrace{[\varphi(X_{s-} + g(X_{s-}, z)) - \varphi(X_{s-})]}_{\Big(\varphi(0)-\varphi(1)\Big)\mathbb{1}_{\{X_s \leq 1\}}} N(ds, dz).$$

Taking the expectation:

$$\mathbb{E}\left[\varphi(X_t)\right] = \mathbb{E}\left[\varphi(X_0)\right] + \mu \int_0^t \mathbb{E}\left[\varphi(X_s)\Big(b(X_s) + \beta \cdot pe'(s)\Big)\right] ds + \frac{\mu^2}{2}\int_0^t \mathbb{E}\left[\varphi(X_s)\right] ds +$$
$$\Big(\varphi(0) - \varphi(1)\Big)e(t).$$

Defining $F_\mu(t) := \mathbb{E}[\varphi(X_t)]$, we can rewrite the above expression as:

$$F_\mu(t) = F_\mu(0) + \mu \int_0^t \mathbb{E}\left[\varphi(X_s)\Big(b(X_s) + \beta \cdot pe'(s)\Big)\right] ds + \frac{\mu^2}{2}\int_0^t F_\mu(s)ds +$$
$$\Big(\varphi(0) - \varphi(1)\Big)e(t).$$

Then, using the hypotesis on b, we get that

$$F_\mu(t) \geq F_\mu(0) + \mu \int_0^t \mathbb{E}\left[\varphi(X_s)(\beta \cdot pe'(s) - \lambda X_s)\right] ds + \frac{\mu^2}{2}\int_0^t F_\mu(s)ds + \Big(\varphi(0) - \varphi(1)\Big)e(t)$$
$$\geq F_\mu(0) + \mu \int_0^t (\beta \cdot pe'(s) - \lambda)F_\mu(s)ds + \frac{\mu^2}{2}\int_0^t F_\mu(s)ds + \Big(\varphi(0) - \varphi(1)\Big)e(t),$$

that is

$$F_\mu(t) \geq F_\mu(0) + \int_0^t \mu\left(\beta \cdot pe'(s) - \lambda + \frac{\mu}{2}\right) F_\mu(s)ds + \Big(\varphi(0) - \varphi(1)\Big)e(t). \tag{2.4}$$

Define $\tilde{\lambda}$ as

$$\tilde{\lambda} := \frac{\varphi(1) - \varphi(0)}{\mu\beta p}$$

and let's choose μ such that

$$-\lambda + \frac{\mu}{2} > 0.$$

We can now proceed by stating that:

$$F_\mu(t) \geq F_\mu(0) + \int_0^t \mu\beta e'(s)F_\mu(s)ds - \tilde{\lambda}\beta\mu e(t)$$
$$\geq F_\mu(0) + \int_0^t \mu\beta e'(s)[F_\mu(s) - \tilde{\lambda}]ds.$$

In differential form:

$$\frac{d}{dt}F_\mu(t) \geq \mu\beta e'(t)\Big[F_\mu(t) - \tilde{\lambda}\Big]. \tag{2.5}$$

Claim: If $F_\mu(0) \geq \tilde{\lambda}$, then $F_\mu(t) \geq \tilde{\lambda}$.
This simply comes from an application of Gronwall Lemma to the function $\tilde{F}_\mu(t) := \tilde{\lambda} - F_\mu(t)$.

Coming back to (2.4) we get that

$$F_\mu(t) \geq F_\mu(0) + \int_0^t \mu\left(\beta e'(s) - \lambda + \frac{\mu}{2}\right)F_\mu(s)ds + \tilde{\lambda}\beta\mu e(t)$$
$$\geq F_\mu(0) + \int_0^t \mu\beta e'(s)F_\mu(s)ds + \int_0^t \mu\left(-\lambda + \frac{\mu}{2}\right)F_\mu(s)ds - \tilde{\lambda}\beta\mu e(t)$$
$$= F_\mu(0) + \tilde{\lambda}\beta\mu e(t) + \int_0^t \mu\left(-\lambda + \frac{\mu}{2}\right)F_\mu(s)ds - \tilde{\lambda}\beta\mu e(t)$$
$$= F_\mu(0) + \int_0^t \mu\left(-\lambda + \frac{\mu}{2}\right)F_\mu(s)ds.$$

In differential form:

$$\frac{d}{dt}F_\mu(t) \geq \mu\left(-\lambda + \frac{\mu}{2}\right)F_\mu(t),$$

which implies that:

$$F_\mu(t) \geq \exp\left(\mu\Big(-\lambda + \frac{\mu}{2}t\Big)\right)F_\mu(0). \tag{2.6}$$

But we know that

$$F_\mu(t) \leq \exp(\mu). \tag{2.7}$$

From (2.6) and (2.7) we get a contradiction. □

Remark 2. *In summary, fixed an (α, p) the phenomena of blow up holds for initial condition that are concentrated around the threshold, namely for initial condition such that*

$$\exists \mu > 2\lambda \text{ such that } F_\mu(0) \geq \tilde{\lambda} \text{ with } \tilde{\lambda} = \frac{\phi(1) - \phi(0)}{\mu\beta p}$$

Assuming for simplicity that the initial condition is deterministic, we get:

$$F_\mu(0) = \exp(\mu x_0)$$

So, given a deterministic initial condition, we look for all p such that:

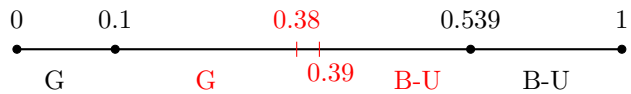
$$p \geq \inf_{\mu \geq 2\lambda} \frac{e^\mu - 1}{\mu\alpha e^{\mu x_0}} := p_c^+(x_0)$$

In figure 2 we present the upper bound for the threshold p_c^+, varying on the parameter of the drift λ and the initial condition x_0.

commenta il plot?

2.1.3 Blow up discussion for the Brownian case

Let's focus on a particular case, in absence of drift, so $\lambda = 0$, with deterministic initial condition $x_0 = 0.8$ and $\beta = 1$. From the previous remark, we get that $p_c^+ \sim 0.539$. Moreover by Theorem 5.2 we get that $p_c^- \sim 0.1$. Through simulations, we look for the critical threshold $p_c(x_0)$ $[p_c^-(x_0), p_c^+(x_0)]$, observing that the threshold is around 0.34. (In red the result obtained by numerics)



In the more general case, where $\alpha \neq 1$, we get that the threshold for p is simply rescaled by the factor α.

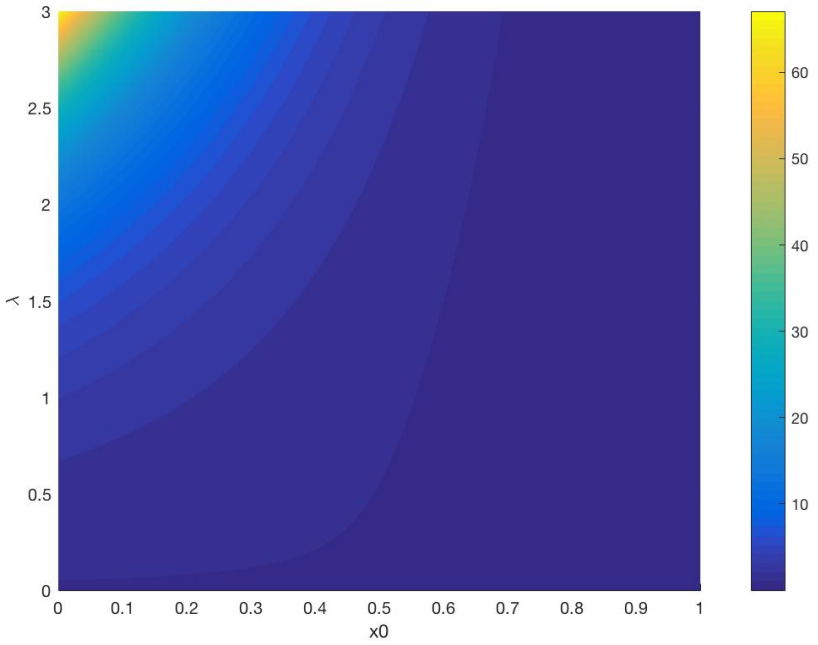


Figure 2.1: Plot of the upper bound estimate p_c^+ for the threshold p_c

2.2 Dependent Random Weights

In this section, we will present a short analysis on the behaviour of the network of neurons, when the connections are dependent. In particular, we will focus on two different kind of interactions, due to random connections. First, we will focus on the case when $\alpha^i\alpha^j$, where $(\alpha^i)_{i=1,\dots,N}$ are independent Bernoullian r.v.(s) with the same distribution, then :

$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \frac{\alpha^i}{N}\sum_{j=1}^{N}\alpha^j M_t^j - M_t^i + W_t^i. \quad i = 1, \cdots, N \tag{2.8}$$

Our first purpose is to find out the limit equation. We conjecture that the limit equation for this system is the following:

$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha\mathbb{E}[\alpha M_t] - M_t + W_t, \tag{2.9}$$

where α is another Bernoullian r.v. with the same distribution as all the α^i; or if the limit equation takes the form

$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha\mathbb{E}[\alpha]\mathbb{E}[M_t] - M_t + W_t. \tag{2.10}$$

with the same hypothesis on α.Our guess is that the limit equation is (2.9), because of the fact that the processes X^i and M^i should not become independent of α^i, even if the network size tends to infinity. So, we shouldn't be able to observe the term $\mathbb{E}[\alpha]\mathbb{E}[M_t]$, but we should see a term like $\mathbb{E}[\alpha M_t]$ instead.

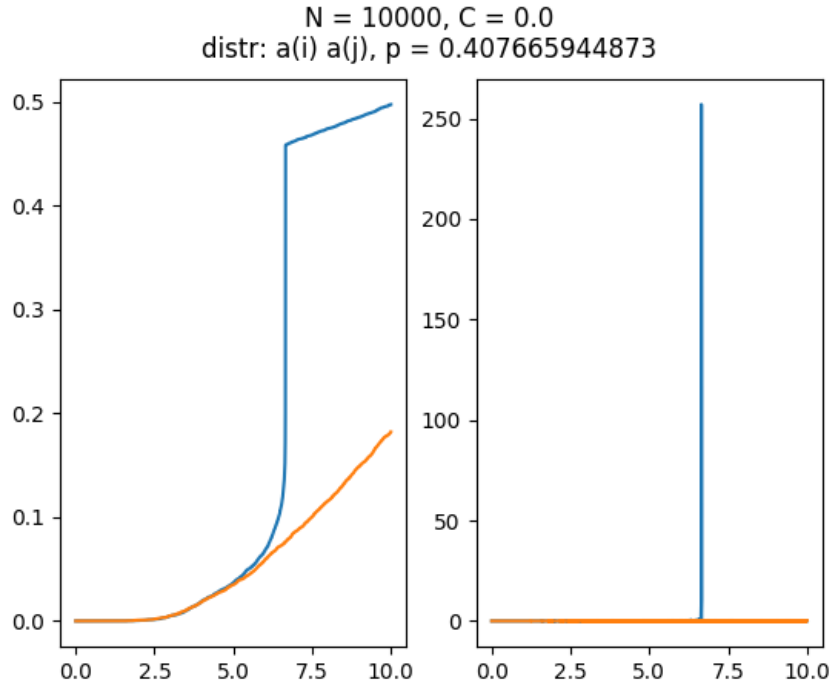
Moreover, heuristically, the system (2.12) define a a network that consist of two different component: the first is a complete sub-network, composed by neurons all connected beetwen themselves, so neurons of the subnetwork feel all the neurons of the subnetwork. The second component is made of isolated neurons. So we expect at the limit equation for average behaviour a randomness, to catch out the duality of the population and moreover we expect that the weight of the kick is calibrated by size of the subnetwork:

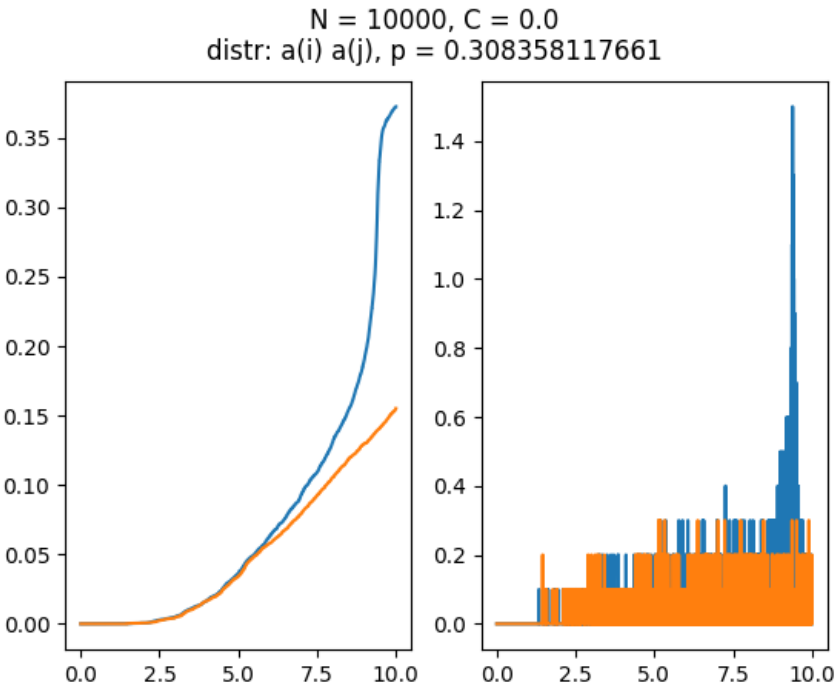
$$\alpha\mathbb{E}\Big[\alpha M_t\Big],$$

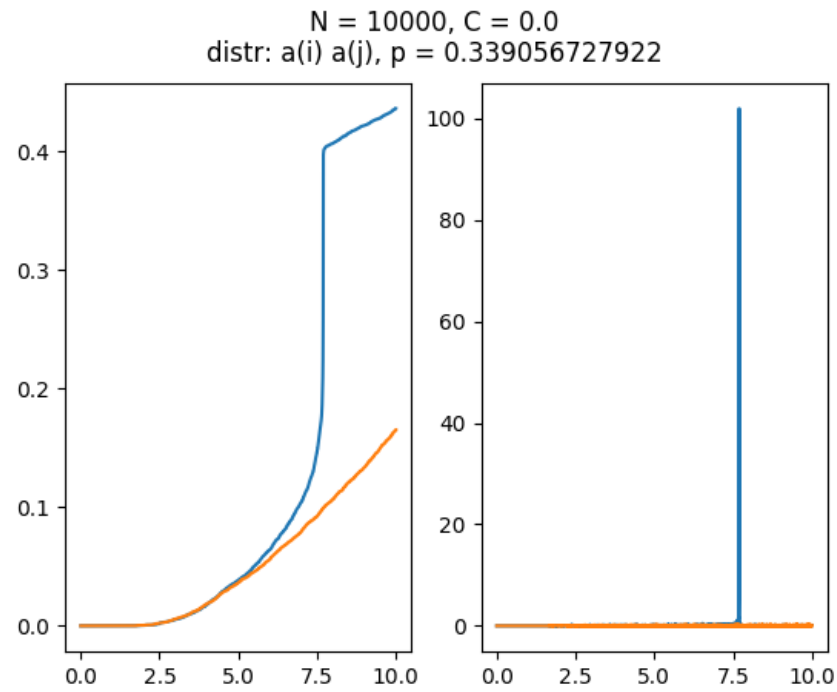
The equation (2.10) can describe the limit equation for a newtwork of neurons, that still present two different population: one of isolated neurons and the other of neurons that interact with all the other neurons, with probability p. So with interaction term:

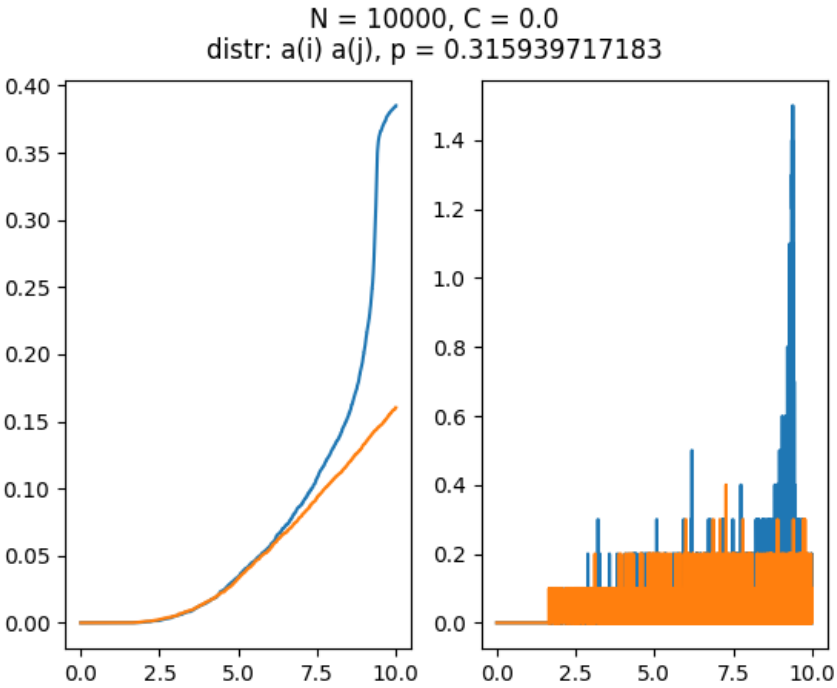
$$\frac{\alpha^i}{N}\sum_{j=1}^{N}\alpha^j M_t^j$$

Numerically, if the limit equation were (2.10), then the interaction term $\frac{\alpha^i}{N}\sum_{j=1}^{N}\alpha^j M_t^j$ should behave, for N large enough, like $\frac{\alpha^i\cdot p}{N}\sum_{j=1}^{N}M_t^j$. We therefore compare numerically those two terms.









We observe a quite different behaviour. Explain d

Remark 3. *Remembering the natural splitting of the network explained just above, we get that, considering the limit of (2.9)*

$$\alpha\mathbb{E}\big[\alpha M_t\big] = \alpha\mathbb{E}\left[M_t \cdot \mathbb{1}_{\{\alpha=1\}}\right] = \alpha p \cdot \mathbb{E}\big[M_t \,|\, \alpha = 1\,\big].$$

Regarding the limit form of (2.10):

$$\begin{aligned} \alpha p \cdot \mathbb{E}\big[M_t\big] &= \alpha p\Big(\mathbb{E}\left[M_t \,|\mathbb{1}_{\{\alpha=1\}}\right] \cdot p + \mathbb{E}\left[M_t \,|\mathbb{1}_{\{\alpha=0\}}\right] \cdot (1-p)\Big) \\ &= \alpha p^2 \cdot \mathbb{E}\Big[M_t \,|\, \alpha = 1\,\Big] + \alpha p(1-p) \cdot \mathbb{E}\Big[M_t \,|\, \alpha = 0\,\Big] \end{aligned} \tag{2.11}$$

CLAIM: *the leading term for blow up in (2.11) is*

$$\alpha p^2 \cdot \mathbb{E}\Big[M_t \,|\, \alpha = 1\,\Big],$$

So we can expect that the threshold for blow up for (2.9), $p_c^2 \approx \sqrt{p_c^1}$

2.2.1 Derivation of the limit equation in the simpler model for $\alpha_{i,j} = \alpha_i \alpha_j$

We now study the limit behaviour of

$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \frac{\alpha^i}{N}\sum_{j=1}^N \alpha^j M_t^j - M_t^i + W_t^i$$

(where α^i are i.i.d. Bernoullian) from a theoretical point of view. Particularly, we consider the simpler model already introduced before, thet is the one described by

$$dX_t^i = \frac{\alpha^i}{N}\sum_{j=1}^N \alpha^j X_t^j dt + dW_t^i.$$

We also introduce, following the notation already used, the processes $\bar{X}^i$ described by

$$d\bar{X}_t^i = \alpha^i \mathbb{E}\left[\alpha^i \bar{X}_t^i\right] dt + dW_t^i.$$

Assuming that X^i and $\bar{X}^i$ have the same initial conditions, we get

$$X_t^i - \bar{X}_t^i = \int_0^t \frac{\alpha^i}{N}\sum_{j=1}^N \alpha^j \left(X_s^j - \bar{X}_s^j\right)ds + \int_0^t \left(\frac{\alpha^i}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \alpha^i \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right) ds.$$

Multiplying by α^i, summing over i and dividing by N, we can assert that

$$\frac{1}{N}\sum_{i=1}^N \alpha^i \left|X_t^i - \bar{X}_t^i\right| \leq \int_0^t \left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2\right)\left(\frac{1}{N}\sum_{i=1}^N \alpha^i \left|X_s^i - \bar{X}_s^i\right|\right) ds + \\ + \int_0^t \left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2\right)\left|\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right| ds.$$

Defining $\tilde{\delta}_t = \frac{1}{N}\sum_{i=1}^N \alpha^i \left|X_t^i - \bar{X}_t^i\right|$, we can write that

$$\tilde{\delta}_t \leq \left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2\right)\left[\int_0^t \tilde{\delta}_s ds + \int_0^t \left|\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right| ds\right].$$

By using Gronwall's lemma:

$$\tilde{\delta}_t \leq \left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2\right)\exp\left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2 t\right)\int_0^t \left|\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right| ds.$$

Also defining $\delta_t = \mathbb{E}[\tilde{\delta}_t]$, $A = \left(\frac{1}{N}\sum_{i=1}^N (\alpha^i)^2\right)$, taking the expectation and using the Cauchy-Schwarz inequality, we have

$$\delta_t \leq \left(\mathbb{E}\left[A^2 \exp\left(2At\right)\right]\right)^{1/2} \int_0^t \left(\mathbb{E}\left[\left(\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right)^2\right]\right)^{1/2} ds.$$

We can manage the last term in the expression above as we did before. In fact:

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j - \mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{N}\sum_{j=1}^N \alpha^j \bar{X}_s^j\right)^2\right] - \left(\mathbb{E}\left[\alpha^i \bar{X}_s^i\right]\right)^2.$$

Expanding the first term in the last line above, we then get

$$\frac{1}{N^2}\mathbb{E}\left[\sum_{j=1}^{N}\left(\alpha^j\bar{X}_s^j\right)^2+\sum_{j\neq k}\alpha^j\alpha^k\bar{X}_s^j\bar{X}_s^k\right]-\left(\mathbb{E}\left[\alpha^i\bar{X}_s^i\right]\right)^2=$$

$$=\frac{1}{N}\left(\mathbb{E}\left[\left(\alpha^i\bar{X}_s^i\right)^2\right]-\left(\mathbb{E}\left[\alpha^j\bar{X}_s^i\right]\right)^2\right)$$

Therefore

$$\delta_t\leq\left(\mathbb{E}\left[A^2\exp\left(2At\right)\right]\right)^{1/2}\cdot\frac{1}{N^{1/2}}\int_0^t\left(\mathbb{E}\left[\left(\alpha^i\bar{X}_s^i\right)^2\right]-\left(\mathbb{E}\left[\alpha^j\bar{X}_s^i\right]\right)^2\right)^{1/2}ds.$$

The derivation of equation (2.10) from the network of interacting neurons:

$$X_t^i=X_0^i+\int_0^t b(X_s^i)ds+\frac{\alpha^i}{N}\sum_{j=1}^{N}\alpha^jM_t^j-M_t^i+W_t^i.\quad i=1,\cdots,N\tag{2.12}$$

adapted to the case of the simple model, follows from a very similar argument.

argomenta meglio

2.2.2 Blow-up Argument

Theorem 4. *Assume that drift coefficient satisfies*

$$b(v)\geq-\lambda v,\quad \textit{for all } -\infty<v\leq 1.$$

If the initial condition is concentrated around the threshold 1*, there is no global in time solution of the SDE*

$$X_t=X_0+\int_0^t b(X_s)ds+\alpha p\cdot\mathbb{E}\left[M_t\right]+W_t-M_t,\tag{2.13}$$

with $\alpha\sim\mathcal{B}(p)$ *and deterministic initial condition* $X(0)=X_0<1$.

Proof. Taking the function $\varphi_\mu(x)=\exp(\mu x)$ for some $\mu>0$ and using the Ito-formula, we have that

$$\varphi_\mu(X_t)=\varphi_\mu(X_0)+\int_0^t\mu\varphi(X_s)\Big(b(X_s)+\alpha pe'(s)+\frac{\mu}{2}\Big)ds+$$

$$+\int_0^t\mu\varphi(X_s)dW_s+\int_0^t(\varphi(X_{s-}-1)-\varphi(X_{s-}))dM_s.$$

Particularly, the last term reduces to $\big(\varphi(0)-\varphi(1)\big)\cdot M_t$, which is negative. Multiplying the equation above by α, recalling that $b(v)\geq-\lambda v$ by hypotesis and that $X_t\leq 1$, we then get

$$\alpha\varphi_\mu(X_t)=\alpha\varphi_\mu(X_0)+\int_0^t\mu\alpha\varphi(X_s)\Big(-\lambda+\alpha pe'(s)+\frac{\mu}{2}\Big)ds+$$

$$+\int_0^t\mu\alpha\varphi(X_s)dW_s+\big(\varphi(0)-\varphi(1)\big)\cdot\alpha M_t.$$

The stochastic integral above remains a martingale, even after multiplying by α, which implies that, taking the expectation and calling $F_\mu^\alpha(t):=\mathbb{E}\Big[\alpha\varphi_\mu(X_t)\Big]$:

$$F_\mu^\alpha(t)=F_\mu^\alpha(0)+\int_0^t\mathbb{E}\left[\mu\alpha\varphi(X_s)\Big(-\lambda+\alpha pe'(s)+\frac{\mu}{2}\Big)\right]ds+\Big(\varphi(0)-\varphi(1)\Big)\mathbb{E}[\alpha M_t].$$

The random variable $\alpha\in\{0,1\}$ and M_t is non-negative, which makes it possible to state that $\mathbb{E}\big[\alpha M_t\big]\leq\mathbb{E}\big[M_t\big]$. Therefore, since $\big(\varphi(0)-\varphi(1)\big)$ is negative,

$$F_\mu^\alpha(t)\geq F_\mu^\alpha(0)+\int_0^t\mu\left(-\lambda+\frac{\mu}{2}\right)F_\mu^\alpha(s)ds+\int_0^t\mu pe'(s)F_\mu^\alpha(s)ds+\Big(\varphi(0)-\varphi(1)\Big)e(t).\tag{2.14}$$

We now assume that

$$-\lambda+\frac{\mu}{2}>0,$$

so that

$$F_\mu^\alpha(t) \geq F_\mu^\alpha(0) + \int_0^t \mu p e'(s) F_\mu^\alpha(s) ds + \Big(\varphi(0) - \varphi(1)\Big) e(t);$$

calling $\tilde{\lambda} := \dfrac{1}{\mu p}\big(-\varphi(0) + \varphi(1)\big)$, we end up with

$$F_\mu^\alpha(t) \geq F_\mu^\alpha(0) + \int_0^t \mu p e'(s) \Big(F_\mu^\alpha(s) - \tilde{\lambda}\Big) ds.$$

Hence, if $F_\mu^\alpha(0) \geq \tilde{\lambda}$, then

$$F_\mu^\alpha(t) \geq \tilde{\lambda}.$$

Starting again from (2.2.2):

$$F_\mu^\alpha(t) \geq F_\mu^\alpha(0) + \int_0^t \mu \left(-\lambda + \frac{\mu}{2}\right) F_\mu^\alpha(s) ds,$$

which implies that

$$F_\mu^\alpha(t) \geq F_\mu^\alpha(0) \exp\left(\mu\Big(-\lambda + \frac{\mu}{2}\Big)t\right).$$

This is a contradiction because $X_t \leq 1$ and $\alpha \in \{0, 1\}$, which means that

$$F_\mu^\alpha(t) \leq e^\mu.$$

□

Remark 5. *As explained in Remark 2, ther is blow-up when*

$$\exists \mu > 2\lambda \text{ such that } F_\mu(0) \geq \tilde{\lambda} \text{ with } \tilde{\lambda} = \frac{\phi(1) - \phi(0)}{\mu p}$$

Assuming for simplicity that the initial condition is deterministic, we get:

$$F_\mu^\alpha(0) = p \cdot \exp(\mu x_0)$$

So, given a deterministic initial condition, we look for all p such that:

$$p \geq \sqrt{\inf_{\mu \geq 2\lambda} \frac{e^\mu - 1}{\mu \alpha e^{\mu x_0}}} := p_c^{+,1}(x_0)$$

In figure ?? we present the upper bound for the threshold $p_c^{+,1}$, varying on the parameter of the drift λ and the initial condition x_0.

Theorem 6. *Assume that drift coefficient satisfies*

$$b(v) \geq -\lambda v, \quad \text{for all } -\infty < v \leq 1.$$

If the initial condition is concentrated around the threshold 1, there is no global in time solution of the SDE

$$X_t = X_0 + \int_0^t b(X_s) ds + \alpha \cdot \mathbb{E}\left[\alpha M_t\right] + W_t - M_t, \tag{2.15}$$

with a deterministic initial condition $X(0) = x_0 < 1$ and $\alpha \sim \mathcal{B}(p)$.

Proof. The strategy is almost the same of the previous case, namely the main object of the proof is $F_\mu^\alpha(t) := \mathbb{E}\Big[\alpha\varphi_\mu(X_t)\Big]$. The identity satisfied by F_μ^α obtained using Ito Formula:

$$F_\mu^\alpha(t) = F_\mu^\alpha(0) + \int_0^t \mathbb{E}\left[\mu\alpha\varphi(X_s)\Big(-\lambda + \alpha e'_\alpha(s) + \frac{\mu}{2}\Big)\right] ds + \Big(\varphi(0) - \varphi(1)\Big)\mathbb{E}[\alpha M_t].$$

with $e_\alpha(t) = \mathbb{E}[\alpha M_t]$. With the same calculation of the previous case, we get

$$F_\mu^\alpha(t) \geq F_\mu^\alpha(0) + \int_0^t \mu \left(-\lambda + \frac{\mu}{2}\right) F_\mu^\alpha(s) ds + \int_0^t \mu e'_\alpha(s) F_\mu^\alpha(s) ds + \Big(\varphi(0) - \varphi(1)\Big) e_\alpha(t).$$

The only difference with respect to the previous case is that in the calculation we have $e_\alpha(t)$ instead of $e(t)$. □

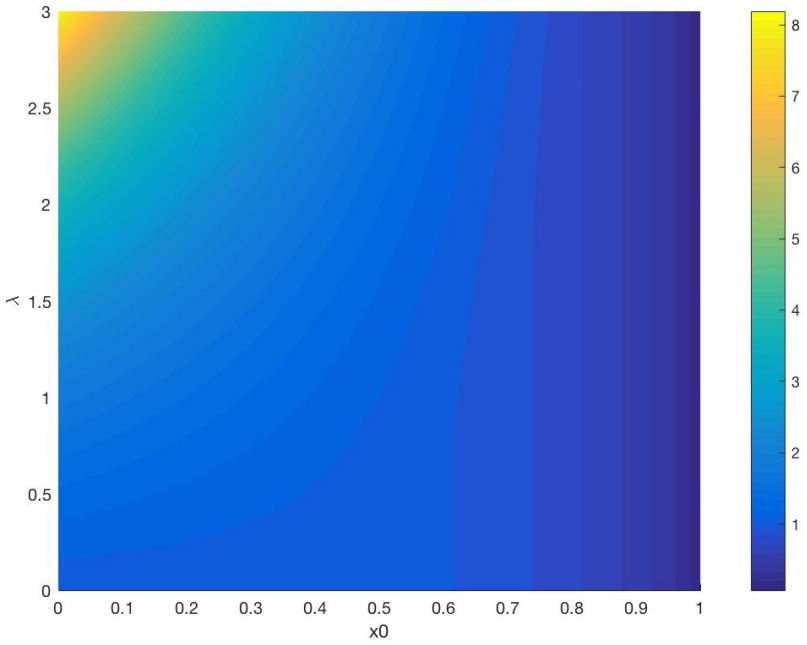


Figure 2.2: Plot of the upper bound estimate $p_c^{+,1}$ for the threshold p_c

Remark 7. *As before, there is blow-up when*

$$\exists \mu > 2\lambda \text{ such that } F_\mu^\alpha(0) \geq \tilde{\lambda} \text{ with } \tilde{\lambda} = \frac{\phi(1) - \phi(0)}{\mu}$$

Assuming for simplicity that the initial condition is deterministic, we get:

$$F_\mu(0) = p \cdot \exp(\mu x_0)$$

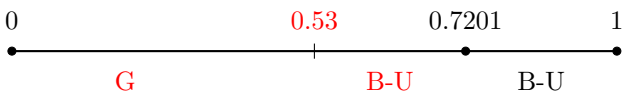
So, given a deterministic initial condition, we look for all p such that:

$$p \geq \inf_{\mu \geq 2\lambda} \frac{e^\mu - 1}{\mu e^{\mu x_0}} := p_c^{+,2}(x_0)$$

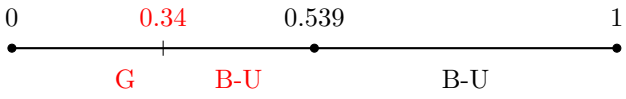
So the upper bound varying on the parameters (x_0, λ) is the same of figure 2.

2.2.3 Blow up discussion for the Brownian case

Let's focus on the case where the network include a non complete subnetwork (2.10) , in absence of drift, so $\lambda = 0$, with deterministic initial condition $x_0 = 0.8$. From the previous remark, we get that $p_c^+ \sim 0.7201$. Through simulations, we look for the threshold value p_c.



Indeed in the case where the network generate a complete subnetwork (2.9), we get



fare simulazioni accurate in $[0.51, 0.55$

fare simulazioni accurate in $[0.3, 0.34]$

2.2.4 Comparison between the two models

We here consider the following SDEs:

$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \frac{\alpha^i}{N} p \sum_{j=1}^N M_t^j + W_t^i - M_t^i,$$
$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \frac{\alpha^i}{N} \sum_{j=1}^N \alpha^j M_t^j + W_t^i - M_t^i,$$

with the usual definitions. We already know that those SDEs converge respectively, as $N \to +\infty$, to the limit SDEs:

$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha p \mathbb{E}\big[M_t\big] + W_t - M_t,$$
$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha \mathbb{E}\big[\alpha M_t\big] + W_t - M_t,$$

again with the usual definitions.

These last SDEs could easily be rewritten as:

$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha p\Big(p \cdot \mathbb{E}\big[M_t \big|\ \alpha = 1\big] + (1-p) \cdot \mathbb{E}\big[M_t \big|\ \alpha = 0\big]\Big) + W_t - M_t,$$
$$X_t = X_0 + \int_0^t b(X_s)ds + \alpha\Big(p \cdot \mathbb{E}\big[M_t \big|\ \alpha = 1\big]\Big) + W_t - M_t.$$

Assuming, as usual, a deterministic initial condition $X_0 = x_0$ and a function $b = 0$, we finally get:

$$X_t = x_0 + \Big(\alpha p^2 \cdot \mathbb{E}\big[M_t \big|\ \alpha = 1\big] + \alpha p(1-p) \cdot \mathbb{E}\big[M_t \big|\ \alpha = 0\big]\Big) + W_t - M_t, \tag{2.16}$$
$$X_t = x_0 + \Big(\alpha p \cdot \mathbb{E}\big[M_t \big|\ \alpha = 1\big]\Big) + W_t - M_t. \tag{2.17}$$

We now study equation (2.16) and compare it to (2.17). First, let us consider the term $\mathbb{E}\big[M_t \big|\ \alpha = 0\big]$: this is the expectation of the process M_t when we assume no interaction. Therefore, we may identify $\mathbb{E}\big[M_t \big|\ \alpha = 0\big]$ with $\mathbb{E}\big[\tilde{M}_t\big]$, where $\tilde{M}_t$ solves the SDE:

$$\tilde{X}_t + \tilde{M}_t = x_0 + \tilde{W}_t, \qquad \tilde{M}_t = \left\lfloor \left(\sup_{[0,t]} \left(\tilde{X}_s + \tilde{M}_s \right) \right)_+ \right\rfloor.$$

Hence,

$$\tilde{X}_s + \tilde{M}_s = x_0 + \tilde{W}_s,$$

which means that

$$\mathbb{E}\left[\tilde{M}_t\right] = \mathbb{E}\left[\left\lfloor \left(x_0 + \sup_{[0,t]} \tilde{W}_s \right)_+ \right\rfloor \right].$$

Calling f_t the density of the running maximum of Brownian motion until time t, we get that

$$\mathbb{E}\left[\tilde{M}_t\right] = \int_{\mathbb{R}} \left\lfloor \big(x_0 + x\big)_+ \right\rfloor f_t(x)dx.$$

Recall that $f_t(x) = \sqrt{2/\pi t}\ \exp\big(-x^2/2t\big)$, so that the expression above becomes

$$\begin{aligned}
\mathbb{E}\left[\tilde{M}_t\right] &= \sqrt{\frac{2}{\pi t}} \int_{\mathbb{R}} \left\lfloor \big(x_0 + x\big)_+ \right\rfloor \ \exp\left(-\frac{x^2}{2t}\right) dx \\
&= \sqrt{\frac{2}{\pi t}} \int_{-x_0}^{+\infty} \left\lfloor \big(x_0 + x\big)_+ \right\rfloor \ \exp\left(-\frac{x^2}{2t}\right) dx \\
&= \frac{4}{\sqrt{\pi}} \int_{-x_0/\sqrt{2t}}^{+\infty} \left\lfloor \Big(x_0 + \sqrt{2t}x\Big)_+ \right\rfloor \ \exp\left(-x^2\right) dx \\
&= \frac{4}{\sqrt{\pi}} \sum_{k \in \mathbb{N}} k \cdot \int_{(-x_0+k)/\sqrt{2t}}^{(-x_0+k+1)/\sqrt{2t}} \exp\left(-x^2\right) dx.
\end{aligned}$$

Particularly, calling $e_0(t) := \mathbb{E}\left[\tilde{M}_t\right]$, we see that the function e_0 is differentiable and its derivative is

$$e_0'(t) = \frac{1}{t}\sqrt{\frac{2}{\pi t}}e^{-\frac{x_0^2}{2t}} \sum_{k\in\mathbb{N}} k\left[(-x_0+k)e^{\frac{-k^2+2x_0k}{2t}} - (-x_0+k+1)e^{\frac{-(k+1)^2+2x_0(k+1)}{2t}}\right].$$

Remembering that $x_0 < 1$, so that $-k^2 + 2x_0k \geq 0$ whenever $k \geq 2$, we obtain the following:

$$e_0'(t) = \frac{1}{t}\sqrt{\frac{2}{\pi t}}e^{-\frac{x_0^2}{2t}}\left[(1-x_0)\left[e^{\frac{-1+2x_0}{2t}} - e^{\frac{-2(1-x_0)}{t}}\right] + \right.$$
$$\left. + \sum_{k\geq 2} k\left[(-x_0+k)e^{\frac{-k^2+2x_0k}{2t}} - (-x_0+k+1)e^{\frac{-(k+1)^2+2x_0(k+1)}{2t}}\right]\right];$$

also observing that $e^{-1/t}$ is increasing in t, we could bound $e_0'(t)$ with:

$$\frac{1}{t}\sqrt{\frac{2}{\pi t}}e^{-\frac{x_0^2}{2t}}\left[C_{x_0,T} + \sum_{k\geq 2} k\left[(-x_0+k)e^{\frac{-k^2+2x_0k}{2T}} - (-x_0+k+1)e^{\frac{-(k+1)^2+2x_0(k+1)}{2T}}\right]\right],$$

that is, increasing the constant $C_{x_0,T}$ if necessary,

$$e_0'(t) \leq C_{x_0,T}\frac{1}{t\sqrt{t}}e^{-\frac{x_0^2}{2t}}.$$

We now come back to (2.16) and (2.17).
Particularly, we can rewrite (2.16) in the form:

$$X_t = x_0 + \alpha p(1-p)\int_0^t e_0'(s)ds + \left(\alpha p^2 \cdot \mathbb{E}\big[M_t\big|\ \alpha = 1\big]\right) + W_t - M_t. \tag{2.18}$$

With respect to (2.17), we observe an additional (random) drift term and an additional multiplicative constant p in the interaction term.

2.3 Independent Bernoullian R.V.(s) with parameter p_N

Considering the network where the synaptic weights are given by the family of i.i.d. Bernoullian random variables $(\alpha^{i,j})_{i,j} \sim \mathcal{B}(p_N)$, we want to compare numerically the behaviour of the interaction term

$$\frac{c}{N \cdot p_N}\sum_{j=1}^{N} \alpha^{ij} M_t^j \tag{2.19}$$

with the interaction term

$$\frac{1}{N}\sum_{j=1}^{N} \beta^{i,j} M_t^j, \tag{2.20}$$

now $(\beta^{i,j})_{i,j} \sim \mathcal{B}(c)$ are independent and identically distributed.
We observe that (2.20) has the same limit behaviour as

$$\frac{c}{N}\sum_{j=1}^{N} M_t^j \tag{2.21}$$

and, given the computational advantage of using this form, we will compare (2.19) to (2.21).

We deal with the neuronal network described by the equation

$$X_t^i = X_0^i + \int_0^t b(X_s^i)ds + \frac{\beta}{p_N \cdot N}\sum_{j=1}^{N} \alpha_N^{i,j} M_t^j + W_t^i - M_t^i,$$

where $\alpha_N^{i,j}$ are Bernoullian random variables with parameter p_N dependent on N, the total number of neurons in the network. We are interested in three cases:

- $p_N \cdot N = \log^{1/2}(N)$;
- $p_N \cdot N = \log(N)$;
- $p_N \cdot N = \log^2(N)$.

Particularly, a distinctive characteristic of this network is that it is possible to have a neuron spiking more than once at the same instant of time.
This can happen for three reasons.
First, it is possible that one neuron receives a cumulative kick by all the other particles which is greater than 1: so the potential of a neuron can jump from a potential less than 1 to a potential greater than 2. We will however ignore this behaviour, considering it as a single spike.
Secondly, it is possible that a neuron spikes, that its kick makes other particles spike and that those, in turn, make the first particle spike again. This behaviour may repeat again: when it repeats just for a finite number of times, we call it a **"finite cascade"**. The third and last possibility is that the cascading behaviour just described goes on infinitely many times: we call it an **"infinite cascade"**. This happens when there are some particles spiking which mutually reinforce each other: **it can be shown that those particles have to be among those with a degree bigger than $\lceil p_N N/\beta \rceil$ FALSE!!!**. Let's give an estimate for the probability of multiple spikes. We here find an upper bound on the probability that a neuron has degree bigger than $\lceil p_N N/\beta \rceil$. We are therefore interested in computing

FALSE

$$\mathbb{P}\left[\frac{1}{N}\sum_{j=1}^{N}\alpha_N^{1,j} \geq \frac{p_N}{\beta}\right],$$

where the apex 1 may be substituted by any other index i.

Observing that p_N/N is bigger than p_N and using the Cramer's Theorem (in the context of Large Deviation Theory), we find out that

$$\mathbb{P}\left[\frac{1}{N}\sum_{j=1}^{N}\alpha_N^{1,j} \geq \frac{p_N}{\beta}\right] \leq e^{-N\cdot\Lambda^*\left(\frac{p_N}{N}\right)}, \tag{2.22}$$

where

$$\Lambda^*(x) := x\log\left(\frac{x}{p_N}\right) + (1-x)\log\left(\frac{1-x}{1-p_N}\right).$$

Remark 8 (Estimate on the upper bound for the probability to have a multiple spikes)**.** *Taking the logarithm and multiplying by -1 the term $e^{-N\cdot\Lambda^*\left(\frac{p_N}{N}\right)}$, we have that*

$$-\log\left(e^{-N\cdot\Lambda^*\left(\frac{p_N}{\beta}\right)}\right) = \frac{1}{\beta}\log\left(\frac{1}{\beta}\right)\log^k(N) + \left(N - \frac{\log^k(N)}{\beta}\right)\log\left(\frac{\beta N - \log^k(N)}{\beta N - \beta\log^k(N)}\right). \tag{2.23}$$

Tha last part above is

$$\left(N - \frac{\log^k(N)}{\beta}\right)\log\left(\frac{\beta N - \log^k(N)}{\beta N - \beta\log^k(N)}\right) = \\ = \left(N - \frac{\log^k(N)}{\beta}\right)\log\left(1 - \frac{1-\beta}{\beta}\frac{\log^k(N)}{N - \log^k(N)}\right).$$

Therefore, the last expression becomes

$$\left(N - \frac{\log^k(N)}{\beta}\right)\left(-\frac{1-\beta}{\beta}\frac{\log^k(N)}{N-\log^k(N)} + o\left(\left(\frac{\log^k(N)}{N-\log^k(N)}\right)^2\right)\right),$$

which can be rewritten as

$$\frac{\beta N-\log^k(N)}{\beta}\left(-\frac{(1-\beta)\log^k(N)}{\beta\big(N-\log^k(N)\big)}\right)+\frac{\beta N-\log^k(N)}{\beta}\cdot o\left(\left(\frac{\log^k(N)}{N-\log^k(N)}\right)^2\right)=$$
$$=\frac{\beta N-\log^k(N)}{\beta N-\beta\log^k(N)}\left(-\frac{(1-\beta)\log^k(N)}{\beta}\right)+$$
$$+\frac{\beta N-\log^k(N)}{\beta N-\beta\log^k(N)}\cdot\frac{N-\log^k(N)}{\log^k(N)}\log^k(N)\cdot o\left(\left(\frac{\log^k(N)}{N-\log^k(N)}\right)^2\right).$$

Finally, we can again rewrite the expression above as

$$\left(\frac{\beta N-\log^k(N)}{\beta N-\beta\log^k(N)}\right)\left(-\frac{1-\beta}{\beta}+o\left(\frac{\log^k(N)}{N-\log^k(N)}\right)\right)\log^k(N).$$

Now, the first term in the expression above converges to 1 *when* $N\to+\infty$*; similarly, the "small o" multiplied by* $\log^k(N)$ *goes to* 0 *when* $N\to+\infty$*; this means that, for every* $c<1$*, definitively,*

$$\left(\frac{\beta N-\log^k(N)}{\beta N-\beta\log^k(N)}\right)\left(-\frac{1-\beta}{\beta}+o\left(\frac{\log^k(N)}{N-\log^k(N)}\right)\right)\log^k(N)\geq -c\left(\frac{1-\beta}{\beta}\right)\log^k(N).$$

Coming back to (2.23):

$$-\log\left(e^{-N\cdot\Lambda^*\left(\frac{p_N}{\beta}\right)}\right)\geq\left[\frac{1}{\beta}\left(\log\left(\frac{1}{\beta}\right)-c\right)+c\right]\log^k(N).$$

Calling $C(\beta):=\frac{1}{\beta}\left(\log\left(\frac{1}{\beta}\right)-c\right)+c$*, we deduce that*

$$e^{-N\cdot\Lambda^*\left(\frac{p_N}{\beta}\right)}\leq e^{-C(\beta)\log^k(N)}.$$

To conclude, we have found that

$$\mathbb{P}\left[\frac{1}{N}\sum_{j=1}^N\alpha_N^{1,j}\geq\frac{p_N}{\beta}\right]\leq e^{-C(\beta)\log^k(N)}.$$

Theorem 9 (Large Deviation Result)**.** *Assuming that* $\beta<1$*,* $p_N=\dfrac{(\log N)^k}{N}$

$$\lim_{N\to\infty}a_N^{-1}\log\left(\mathbb{P}\left[\frac{1}{N}\sum_{j=1}^N\alpha_N^j\geq\frac{p_N}{\beta}\right]\right)=-C(\beta)$$

with $a_N=N\cdot p_N$ *and* $C(\beta):=\dfrac{1}{\beta}\left(\log\left(\dfrac{1}{\beta}\right)-1\right)+1$

Proof. To prove the large deviation result we need an upper bound estimate and a lower bound estimate. The lower bound estimate is the one given in (2.22). The argument for the lower bound is a very classical one, the main tool is given by Cramer Transformation of random variable. dettagli

$$\mathbb{P}\left(\frac{1}{N}\sum_{j=1}^N\alpha_N^j\geq\frac{p_N}{\beta}\right)=\mathbb{P}\left(\frac{1}{N}\sum_{j=1}^N\tilde{\alpha}_N^j\geq\frac{p_N}{\beta}-x\right)$$

where $\tilde{\alpha}_x^j=\alpha^j-x$. Let's consider a family of random variable $\hat{\alpha}^j$ given by Cramer Transformation, so that their CDF is

$$\hat{F}(x)=\frac{1}{A_N}\int_{-\infty}^x e^{\tau y}dF(y)$$

with $A_N=\exp(-N\cdot C(\beta)\cdot p_N)$. First, let us choose x such that $\hat{F}$ is a CDF, so such that

$$A_N=g(\tau)$$

where ϕ is the moment generating function of $\tilde{\alpha}_x^j$:

$$x = -\frac{1}{\tau}\log\left(\frac{e^{-C(\beta)}p_N}{1-p_N+e^{\tau}p_N}\right)$$

Moreover, we impose that the family $\hat{\alpha}^j$ is centered and with positive variance.

$$\mathbb{E}\left[\hat{\alpha}_1\right] = \hat{g}'(0) = \frac{1}{A_N}g'(\tau) = 0 \tag{2.24}$$

where $\hat{g}$ is the moment generating function of $\hat{\alpha}_i$. So we choose

$$\tau = \log\left(\frac{x\cdot(1-p_N)}{(1-x)p_N}\right)$$

namely, the point at which the minimum for g is reached. Through this choice of τ we get that the variance is positive and that it does not depend on N.

$$VAR\left[\hat{\alpha}^j\right] = \hat{g}''(0) = \frac{1}{A}g''(0) = \frac{g''(0)}{g(\tau)} = \frac{G_1(x)}{G_2(x)} \tag{2.25}$$

Let us verify that

$$\mathbb{P}\left(\frac{1}{N}\sum_{j=1}^{N}\alpha_N^j \geq \frac{p_N}{\beta}\right) = (A_N)^N\mathbb{E}\left[\exp(-\tau\cdot\hat{\underline{S}}_N)1_{\hat{\underline{S}}_N \geq N\cdot(\frac{p_N}{\beta}-1)}\right] \tag{2.26}$$

where $\hat{\underline{S}}_N$ states for the empirical average.

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{N}\sum_{j=1}^{N}\tilde{\alpha}_N^j \geq \frac{p_N}{\beta}-x\right) &= \mathbb{P}\left(\tilde{\underline{S}}_N \geq N\left(\frac{p_N}{\beta}-x\right)\right)\\
&= \int_{x_1+\cdots+x_N\geq N\left(\frac{p_N}{\beta}-x\right)} dF_1(x_1)\cdot\cdots\cdot dF_N(x_N)\\
&= A^N\int_{x_1+\cdots+x_N\geq N\left(\frac{p_N}{\beta}-x\right)} e^{-\tau(x_1+\cdots x_N)}d\hat{F}_1(x_1)\cdot\cdots\cdot d\hat{F}_N(x_N)\\
&= (A_N)^N\mathbb{E}\left[\exp(-\tau\cdot\hat{\underline{S}}_N)1_{\hat{\underline{S}}_N\geq N\cdot(\frac{p_N}{\beta}-1)}\right]
\end{aligned}$$

Because the variance (2.3) does not depend on N, we can apply the TLC to prove that

$$\liminf_{N\to\infty}\mathbb{E}\left[\exp(-\tau\cdot\hat{\underline{S}}_N)1_{\hat{\underline{S}}_N\geq N\cdot(\frac{p_N}{\beta}-1)}\right] \geq 0$$

So we can conclude

scrivere b
i conti

$$\begin{aligned}
&\liminf_N a_N^{-1}\log\mathbb{P}\left(\frac{1}{N}\sum_{j=1}^{N}\alpha_N^j \geq \frac{p_N}{\beta}\right)\\
&= \liminf_N a_N^{-1}\log(A_N)^N\mathbb{E}\left[\exp(-\tau\cdot\hat{\underline{S}}_N)1_{\hat{\underline{S}}_N\geq N\cdot(\frac{p_N}{\beta}-1)}\right]\\
&\geq -C(\beta)
\end{aligned}$$

□

Remark 10 (Probability of having a network with no possible cascade behaviour)**.** *We are here interested in computing*

$$\begin{aligned}
\mathbb{P}\left[\exists i\middle|\ \frac{1}{N}\sum_{j=1}^{N}\alpha_N^{i,j} \geq \frac{p_N}{\beta}\right] &= 1-\mathbb{P}\left[\forall i,\ \frac{1}{N}\sum_{j=1}^{N}\alpha_N^{i,j} < \frac{p_N}{\beta}\right]\\
&= 1-\left(\mathbb{P}\left[\frac{1}{N}\sum_{j=1}^{N}\alpha_N^{1,j} < \frac{p_N}{\beta}\right]\right)^N,
\end{aligned}$$

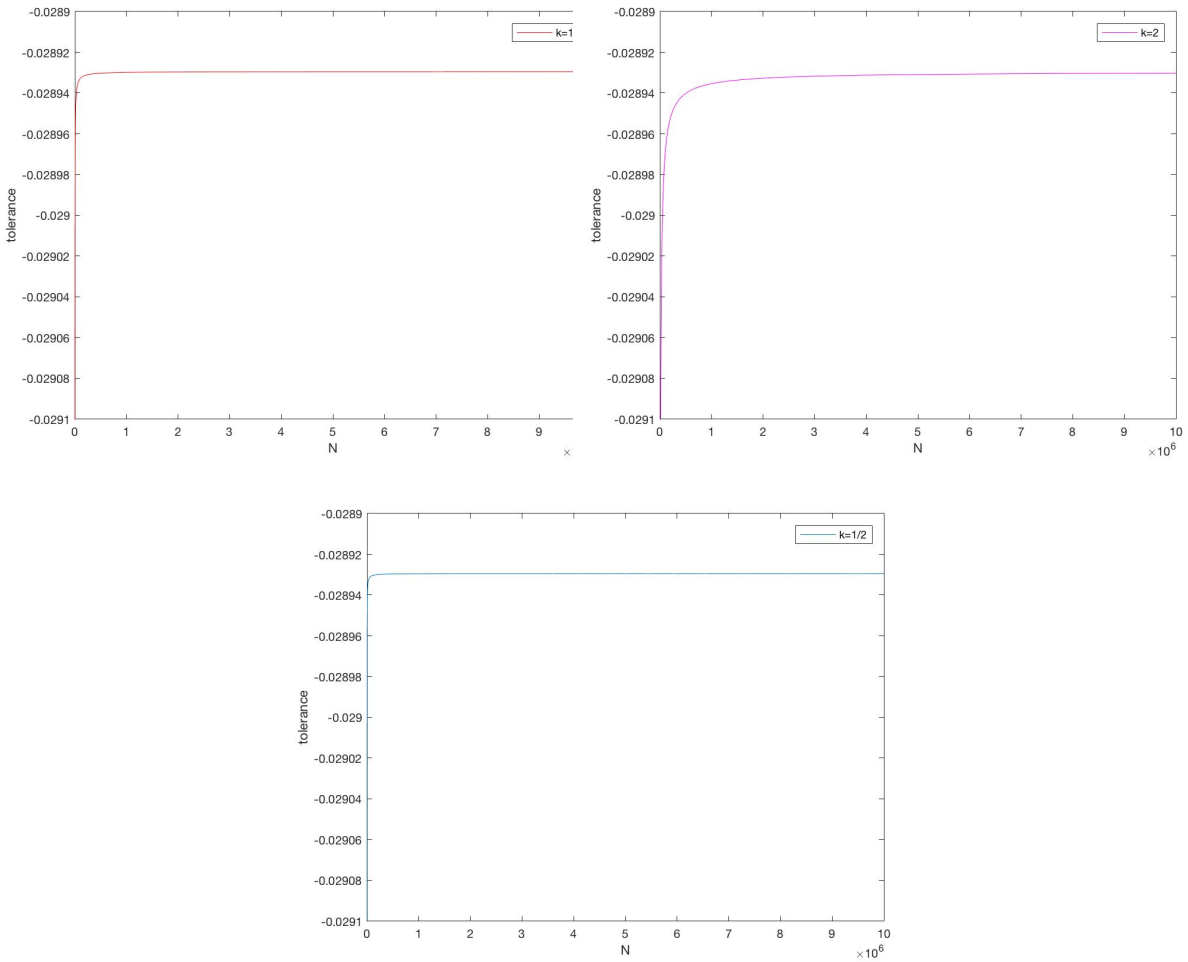


Figure 2.3: Numerical estimation of the constant $C(\beta)$ for different values of k.

because of the independence properties on the random connections $\alpha_N^{i,j}$.

By using the inequality found in the previous paragraph:

$$\mathbb{P}\left[\exists i \middle| \frac{1}{N}\sum_{j=1}^{N}\alpha_N^{i,j} \geq \frac{p_N}{\beta}\right] = 1 - \left(1 - \mathbb{P}\left[\frac{1}{N}\sum_{j=1}^{N}\alpha_N^{1,j} \geq \frac{p_N}{\beta}\right]\right)^N$$
$$\leq 1 - \left(1 - e^{-N\cdot\Lambda^*\left(\frac{p_N}{N}\right)}\right)^N \leq 1 - \left(1 - e^{-C(\beta)\log^k(N)}\right)^N.$$

Chapter 3

Algorithm

In this section we are going to develop an algorithm for the simulation of neurons the potentials of which are modeled with a mean-field equation. In this case the equation is the result of a limit approximation of the behaviour of an infinite network of integrate and fire neurons. In order to obtain relevant simulation results one should consider the means to simulate a network as big as possible. Not only to get closer to the infinite network hypothesis but also because the number of neurons in biological brains is typically huge: a drosophilia for instance has a number of neurons of order 2.55×10^5, while for human beings this number verges on the 90 billions - and it gets even higher for bigger animals like elephants or whales.

3.1 Simulated model

First, we need to discuss about some choices that were made about the model. As a reminder, the potential of the neurons follows the mean-field equation:

$$\forall i \in \{1, \cdots, N\}, V_i(t) = V_i(0) + \int_0^t b(V_i(s))ds + \sigma W_t + \sum_{j=1}^{N} J_{j\to i} M_{j,t} - M_{i,t}(V_{i,T} - V_{i,R}) \tag{3.1}$$

where,

V_i is the membrane potential of the neuron i

$b : \mathbb{R}_+ \to \mathbb{R}_+$ is a drift, and is a lipschitz function of the potential

W_t is a Brownian motion

$M_{i,t} = \sum_k \delta(t - T_{i,k})$, where δ is the Dirac Delta at zero and

$T_{i,k}$ is the times of the k^{th}-spike coming out of the neuron i

$J_{j\to i}$ is the synaptic weight of the connection from j to i

$V_{i,T}$ is the membrane's potential threshold of the neuron i

$V_{i,R}$ is the reset value of the membrane potential after spiking of the neuron i

The $(V_i)_{i\in\{1,\cdots,N\}}$ are discontinuous functions of time, and the discontinuities represent neurons spiking in the system. By simulating this system of neurons we are interested in finding what parameter set creates a given pattern, and more specifically a blow-up of the system, when at a given time the spiking rate tends to infinity.

A spike is an abstract and instantaneous event marking a reset of the potential for the spiking neuron and change in potential for postsynaptic neurons. A neuron i can emit a spike only if its potential is higher than the threshold $V_{i,T}$. The choice has been made to consider the spikes as point processes, with the probability of having a spike depending on the potential of the neuron. Here we must introduce the function $f : \mathbb{R} \to \mathbb{R}_{+,*}$ that will be used to compute the spiking probability. f is defined so that:

$$f : \begin{array}{ll} \mathbb{R} \to & \mathbb{R}_+ \\ x \mapsto & f(x) = C \times \mathrm{Pos}(x - V_T)^p \end{array}$$

where C and p are constants (actually $C >> 1$) and Pos is defined so that:

$$\text{Pos} : \begin{array}{c} \mathbb{R} \to \mathbb{R}_+ \\ x \mapsto Pos(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \end{array}$$

This function is used in the rejection-sampling algorithm explained in section 3.2. It is equal to zero for values of the potential below the threshold, as biological neurons do not spike when their potential is below such a threshold (well, actually it depends on the type of neurons we are dealing with, but this is mostly true for cortex neurons). When the potential exceeds the threshold, the probaility of spiking should quickly increase so that the neuron will quickly spike.

Also an important function is the approximation function $\tilde{\text{f}}$ defined as:

$$\tilde{\text{f}} : \begin{array}{c} \mathbb{R}^3 \to \mathbb{R}_+ \\ (x, [A, B]) \mapsto \tilde{\text{f}}(x, [A, B]) = \max\limits_{x \in [A,B]} (f(x)) \end{array}$$

There are several reasons for the choice of a probability of spiking and not a deterministic condition. First, even if there is still a kind of threshold, this model opens the possibility for simulating neurons with a softer threshold (or even no threshold at all if $\forall i \in \{1, \cdots, N\}, V_{i,T} = 0$). But the most important reason is actually to avoid having no solution at blow-up time because of an infinite amount of spikes at that time. This is interesting for studying the long-term behaviour of such a system.

In short, we are interested in discrete abstract events marquing discontinuities in the neurons' membranes potential. These events appear at random, following a distribution that is function of this membrane potential. These are important considerations, because they make explicit the kind of algorithm we need. From now on, these discontinuities will be named spikes or events indifferently.

The reset value $V_{i,R}$ is the value of the potential of the neuron i just after a spike. It is obviously smaller than the threshold value, but the difference between the two is important because it influences how much time a neuron must wait after a spike for being able to spike again. In biological neurons there is what is called a refractory period, which is not modelled here, during which a neuron cannot spike after having emmited a spike. The potential of a neuron in such state is very low, actually lower than the value of the potential at rest. It is fixed at 0, to be compared to 1 for the threshold.

The synaptic weights $(J_{j \to i})_{(i,j) \in \{1, \cdots, N\}^2}$ are the amount by which the potential of a neuron i children of neuron j changes when j spikes. The $M_{i,t}$ are the history of neurons, that is to say how many times the neuron i has spiked.

Finally the $b : \mathbb{R}_+ \to \mathbb{R}_+$ function makes the neurons naturally proned to spike, as it makes the neurons potentials tend to a given value, higher than the threshold. b is defined as:

$$b : \mathbb{R}_+ \to \mathbb{R} x \mapsto b(x) = -\lambda(x - a), (\lambda, a) \in \mathbb{R}_+^2$$

3.2 Rejection sampling algorithm

Here we present the actual algorithm developed for the simulation of the model presented above. Discussions about its complexity and its actual implementations can be found in sections 3.4 and 3.5.

3.2.1 Clock-driven vs. event-driven algorithm

The classical way of solving a differential equation is to define a time step Δt and then use a numerical method (Runge-Kutta or linear multistep) for recursively solving with $y(t + \Delta t) = f(Y(t))$. These methods usually give good approximations of the exact solution, while being quite simple to implement. They are however time expensive, because of the value Δt that must be small enough for not missing any quick variations of the solution.

Event-driven simulations on the contrary optimises the computations to focus on events, computing the state of the system only when they occur. What are events is an arbitrary choice, but most of the times they represent a change in the system dynamic (variation in the derivative, discontinuities, etc).

As we are going to explore the spiking behaviour of a network of neurons, the second option comes naturally. This method is also very natural when considering a stochastic system, with events like spikes appearing at random with a non common distribution, so that a rejection sampling method is necessary for getting the events.

3.2.2 The algorithm

Algorithm 1 Simulator

N: number of neurons
$\tilde{\mathrm{T}}$: time elapsed since last event in the system
T_{last}: absolute time of the last event in the system
T^i_{next}: absolute time of the nth event of neuron i
$\mathscr{P}$: poisson distribution
$\mathscr{U}$: uniform distribution
$\tilde{\mathrm{f}}$: approximation function of function f ($\tilde{\mathrm{f}}(x) \geq f(x)$)
repeat
  repeat ▷ Steps 0-1
    determine an interval $[A, B[$ on which sampling
    compute the array of $(\tilde{\mathrm{f}}_i(V_i(A)))_{i\in\{1,\cdots,N\}}$ and $\sum_i \tilde{\mathrm{f}}_i(V_i(A))$ on interval $[A, B[$
    $\tilde{\mathrm{T}} \sim \mathscr{P}(\lambda = \sum_i \tilde{\mathrm{f}}_i)$
    $T_{next} \leftarrow T_{last} + \tilde{\mathrm{T}}$
  until in good interval AND at least one $\tilde{\mathrm{f}}_i \neq 0$
  $i \leftarrow \mathbf{argmin}_{i\in\{1,\cdots,N\}} \left(\sum_j \tilde{\mathrm{f}}_j(V_j(A)) * u \sim \mathscr{U}([0,1]) < \tilde{\mathrm{f}}_i(V_i(A)) \right)$ ▷ Step 2
  $T_{i,n} \leftarrow T_{next}$ ▷ Step 3
  $V_i(T_{i,n}) \leftarrow a + e^{-\lambda\tilde{\mathrm{T}}}(V_i(T_{i,n-1}) - a) + \sigma\mathscr{N}(0, \frac{1-e^{-2\lambda\tilde{\mathrm{T}}}}{2\lambda})$
  $\mathbb{P}(\text{accepting spike of neuron i}) = \frac{f_i(V_i(T_{i,n}))}{\tilde{\mathrm{f}}_i(V_i(A))}$ ▷ Step 4
  if spike is accepted **then**
    update potentials of all postsynaptic neurons
until an end condition is met

The algorithm is presented in figure 3.1. It explicits the several steps for simulating the network of neurons.

The main idea here is the rejection sampling method for finding the spikes. This method is applied as many times as necessary for reaching a certain condition, typically a number of spikes found. The actual sampling is made at step 4, but several steps are necessary before being able to sample events.

The first step (0) is looking for a time interval, where the next event shall be looked for. This step is actually important for the speed of the algorithm, as bounding the time also makes possible to derive a better approximation for the approximation functions $\tilde{\mathrm{f}}_i$. The separation between two sequential events follow a Poisson distribution of parameter the sum of the approximation functions (step 1):

$$T_{k+1} = T_k + \tilde{\mathrm{T}}, \text{ with } \tilde{\mathrm{T}} \sim \mathscr{P}(\sum_i \tilde{\mathrm{f}}^i) \tag{3.2}$$

This step gives the time when the event is occuring, and now the neuron responsible for the event must be found. The sum of the approximation functions is used again, as the probability of having a spike depends on them. The bigger the value of $\tilde{\mathrm{f}}_i$, the higher the probability that the neuron i will be the next spiking. A random number, uniformly distributed, choses the winner.

Knowing which neuron is having an event and at what time it is then possible to update its potential $V_i(T_{next})$ (step 3). It is important to remark that if the event was until now categorised as T_{next}, it is now differentiated as an event of neuron i (2 in the figure). An event in the system is indeed linked to the neuron having this event, but the same neuron is usually not responsible for two consecutive events in the system. That is why T_{last} is not equal to $T_{2,n-1}$ in the figure: T_{last} is the last event in the system (caused for instance by neuron 1), while $T_{2,n-1}$ is the last event of the neuron 2.

The value at time $T_{i,n}$ of V_i was necessary for computing the value of the probability function $\mathrm{f}_i(V_i(T_{i,n}))$ and thus classifying the event between false and true spike (step 4). The value of a random variable uniformly distributed between 0 and $\tilde{\mathrm{f}}$ gives the classification: if its lesser than $f(V_i(T_{i,n}))$ the spike is a true one, greater its a false one. If the spike is a true one the potential of the spiking neuron is reset to $V_{i,R}$ while the potentials of all the postsynaptic (children) neurons are updated to the time of event T_{next}. The value of the synatic weight $J^{i\to j}$ is then added to their potential (possibly making them reach their threshold). If the spike was a false one, nothing happens.

The algorithms then loops back to step 0 until a given condition (number of spikes reached, time elapsed, etc.) is met.

3.3 Graph management

Algorithm 2 Generation of a matrix of children

```
RNG: Random Number Generator
p: Probability of connection between two neurons
M_{i,j}: matrix of interaction: equals 1 for a connection, 0 otherwise
RNG.B(p): returns 0 or 1, bernouilli distributed with probability p.
function INTERACTION MATRIX(RNG, p)
    for i ← 1 to n do
        for j ← 1 to n do
            M_{i,j} ←RNG.B(p)
```

Algorithm 2 is the classical algorithm used for creating an Erdös-Renyi directed graph. It is composed of two loops, looking at all pairs (i,j) of neurons. The probability of having a connection between these two neurons (and from i to j) is Bernouilli of a constant parameter p. At each call of the RNG, a random value Bernouilli distributed is thus created and the status of the RNG changes from S_k to S_{k+1}. The result of whether there is a connection or not between two neurons is stored in a matrix called interaction matrix, of values equal to 1 if there is a connection and 0 otherwise. This matrix can be huge in the case where there are a great number of connections, as its size is of order the number of neurons squared. Even if only the relevant parts of the matrix are actually stored (meaning the ones of the matrix), the size is still of order pN^2. That means for instance the graph of a network of $N = 10^5$ neurons needs at maximum 10^{10} storage units (bits, bytes, etc.), which is beyond the amount of memory typically available in most personnal computers nowadays.

So another idea would be to "compress" the data so that it takes less space, and the convenient thing here

Algorithm 3 Generation of a vector of rng states

```
RNG: random Number Generator
p: probability of connection between two neurons
N: number of neurons in the system
S: vector of pairs (Status of the RNG, Number of children of neuron i)
RNG.BIN(N,p): returns a random value following the binomial distribution of parameter N and p
RNG(S_i): returns the index of a child for i among all other unchosen neurons
RNG.STATUS: returns the current internal state of the RNG
I_i: number of children of neuron i
function MAKE VECTOR(RNG, p)
    for i ← 1 to N do
        I_i ←RNG.BIN(N,p)
        S[i] ←(RNG.STATUS,I_i)
        for j ← 1 to I_i do
            RNG(S_{j+∑_{k=1}^{i-1} I_k})
```

is that when looking up closely to the classical generation method, the data of the matrix is already compressed in the state of the RNG at the beginning of the algorithm. The compression rate is great, but usability is really bad, because in order to access to any connection value one would have to reconstruct a great part of the matrix before. A middle way solution is the one presented by algorithm 3, where the idea of using the RNG state as a compression of the interaction matrix can be extended further to compress not the whole matrix but only its rows, or the children of the neurons to say it otherwise.

First, the graph must be generated the same way it is done in the classical method. But instead of storing, for each child j of the neuron i, the value of the connection in a matrix, we just remember that associated to i is the state S_i of the RNG. Then for each child j the value of the connection is generated in a classical way, except for the storing in a matrix part.

The reason why this method is really effective in our case is that whenever a neuron spikes, we have to update the potential of all of its children. The value of the connectivity between two specific neurons is actually never wanted, and so the matrix-based method and the reconstruction-based method will both be used in a similar fashion, that is, for a given neuron i, make something with all its children. Hence the loop on all the

Algorithm 4 Comparison of usage between clasical method and reconstruction

function COMPARISON
INTERACTION MATRIX(RNG_1,p)
MAKE VECTOR(RNG_2,p)
⋮ Simulation
Neuron i is spiking
▷ Using matrix graph
for j ← 1 **to** N **do**
if $\mathscr{M}_{i,j} = 1$ **then**
Update potential of neuron j
▷ Using reconstruction
S[i]=($S_{\sum_{k=1}^{i-1} I_k}, I_i$)
RNG.SETSTATE(S[i][1])
for j ← 1 **to** I_i **do**
Child_j ←RNG($S_{j-1+\sum_{k=1}^{i-1} I_k}$)
Update potential of neuron Child_j

	Time complexity at creation	Time complexity during usage	Space complexity
Reconstruction method	$\mathscr{O}(N \times \max(I_i))$	$\mathscr{O}(\max(I_i))$	$\mathscr{O}(N)$
Interaction matrix	$\mathscr{O}(N^2)$	$\mathscr{O}(1)$	$\mathscr{O}(N^2)$

Table 3.1: Table of time and memory complexity. I_i refers as before to the influencees of the neuron i

children in the comparison algorithm 4.

A question raises then that is the question of the complexity of such a reconstruction-based method compared to the complexity of the classical one. This is a broader matter actually that must be extended to all that have been previously presented here, and so it deserves its own section.

3.4 Complexity: memory and instructions

Definition 11 (Algorithmic complexity)**.** *Analysis of algorithms is the domain interested in determining the amount of resources, may they be time, storage or other resources, necessary in order to execute a given algorithm. Usually the idea is to find a function relating the input of an algorithm to to the number of steps it takes (its time complexity) or the number of storage locations it uses (its space complexity). Most of the time the complexity is examined in the asymptotic sense, in other word, how much of a resource do one need in order to run the algorithm when the input's size grows to infinity? The Big O notation is then used.*

The use of the Big O notation is not insignificant, as this means that two algorithms with the same order of complexity (or two implementations of the same algorithm) may execute with very different amount of resource because there may exist a significant factor between the two (that will become unsignificant asymptotically, but not for small inputs). Knowing this limitation, this kind of analysis focus more on differences of the orders of complexity between two algorithms/implementations.

The first complexity is given as an example of how the complexity is computed. In the classical algorithm for generating an interaction matrix, for all potential parent,one must look at all the other neurons in order to determine whether they are a child of the first one. Hence the double loop on all neurons, for looking at all the elements of the interaction matrix. In the case of the reconstruction method presented above, the number of children of a given parents is determined before hand, and in the second loop their index.

This is the method that is actually used for another type of graph storage structures, called interation list (they have a space complexity similar to interaction matrices, so we are not discussing them here), at it allows to gain some time at the creation and space in memory.

It is interesting to note that as the only input of the algorithm (at least the only asymptotically relevent input) is the number of neurons, everything that is constant in regards to this number does not appear in the complexity. For intance, the random number generation takes time, but as it is constant in regards to the number of neurons it does not appear in the final computation of the complexity.

Finally for the first column, one should take care that even if the time complexity at creation of the reconstruction method is lower than the complexity of the interation matrix in practice, this is not the case theoretically, as one should always look for the worst case scenario, that is the complete graph in this case. In

this scenario, any neuron has all neurons (including self) as children, and so $\max(I_i) = N$.

The interesting parts in this table are the last two columns. The space complexity is obviously where the method shines the most, as it helps gain at least an order of magnitude, making it very relevant for storing very big graphs. On the other hand, the time complexity using the vector of RNG states is worse than the time complexity of using the interaction matrix, but as shown in the example algorithm 4, as we are only interested in manipulating the children (and all of them at the same time), the complexity of using the interaction matrix in our case would have been $\mathcal{O}(N)$ anyway, so the method is very interesting at usage.

The figure 3.3 exposes the differencces in complexity between an event-driven algorithm (in blue) and a clock-driven algorithm (in red). The complexity of clock-driven algorithms is of order N^2. Event-driven algorithms have much less complexity, of order N. But as our algorithm is based on a thinning method, the complexity is higher, but still linear of N.

3.5 Implementation

The algorithm was first prototyped using Java and the final implementation was produced in C. Using higher level languages often poses issues with his kind of problematics as they tend to trade scarcity in resource cnsumption for easiness of use. Also using compiled languages opens the door of compiler optimisations with greatly improved speed and memory.

About memory consumption, the representation of numbers, especially real numbers, is a big issue when dealing with this kind of algorithms. There are indeed both very big and very small numbers used at the same time during the simulation. Summing the values of all the approximation functions (10^5 values) could lead to overflow, that is a lack of bits for correctly representing the number. Different languages uses different ways of overcoming this kind of limitations, depending on which standard is actually implemented. For Java it is the IEEE 754 binary floating point representation, which makes possible to represent floating point numbers with at least 15 digits of precision (not accuracy). For C it really depends on the floating point type chosen.

3.5.1 A word about the rng

The Random Number Generator is a very important component of the algorithms described above and the choice has been made of using the well known Mersenne-Twister, which can provide high quality integer or floating point ranodm numbers. The version we used is the one developped and maintained by the creator of the Mersenne-Twister himself, Makoto Matsumoto, and can be found at http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html.

A related subject is the distribution followed by the random numbers. The algorithm presented earlier needs several random numbers from various distributions, none of them being too exotic (uniform numbers with arbitrary upperbound and normal numbers centered on zero with arbitrary variance), but as the library used only provide uniform random series of 32 or 64 bits and random floating point numbers in [1,2) or [0,1], [0,1), (0,1], (0,1), we must implement a way to generate the numbers drawn from our arbitrary distributions. In both cases a sampling method (again) is used, and we are going to describe them.

First the uniform case. We have a random integer generator, and we assume it gives perfectly random numbers in $0, 1, ..., M$, while we want uniform numbers in $0, 1, ..., U$. A common method to achieve this is to use a modulo: $\mathbb{U}([0, M])mod[U + 1]$. The issue here is that unless M is a multiple of $U + 1$, the uniformity of the numbers is lost using this method. The case in which the method works can be used as a trick for guarrantying uniformity though. Indeed, if M is not a multiple of $U + 1$, there must exist one that is lower than M but close anyway. Knowing that, we can simply take any number that is lower than $U + 1$ and in the rare cases a number greater than $U + 1$ is drawn, we just have to reject it and draw another one.

3.5.2 Improving speed

With the increase of the number of computers having several cores, it becomes more and more interesting to design algorithms with the possibility to use parallelisation for speeding them. Yet, this is not really possible for some of them, and here the main loop of the rejection sampling method exposed in section 3.2 is a good example. Finding the next event time relies on the sum of the approximation functions, which indirectly depend on the potential of each neuron which is computed at the previous event. But if this sequential part cannot be parallelised, some other time consuming parts can!

There is indeed a lot of computations involving large arrays of numbers, be it the computation of the approximation functions for all neurons or the update of the potential of all children neurons. Thanksfully a lot of tools exist that answer this problematic, and one very adapted to our problem is openmp, which is the solution

used for the implementation. This solution is also compatible with the parallelisation of the reconstruction method: instead of storing one state of the RNG per parent neuron, one could sample the state several times during the graph generation. For instance storing the state every 100 children neurons instead of one time. The different states can be used to instanciate several parallel RNGs that will recreate the children in parallel.

There are also solutions for speeding the random number generation by directly generating streams of random numbers instead of on number at a time, again relying on the possibility to parallelise such generation.

Finally, there is one parameter that have not been discussed while having a certain importance on the speed of the algorithm, it is the [A,B[interval, step 0 of the algorithm in figure 3.1. The smaller the length of this interval the closer the approximation function $\tilde{f}$ if of the probability function f. But when the length of the interval is too small, then the time of events, Poisson generated, are greater than the upper-bound of the interval, leading to frequent change in the interval (and the algorithm comes closer to the time complexity of clock-driven algorithms). Finding an optimal here is a difficult issue because the size of the interval depends on the expected value of the Poisson distribution which is the sum of the approximation function computed on the interval [A,B[$\cdots$

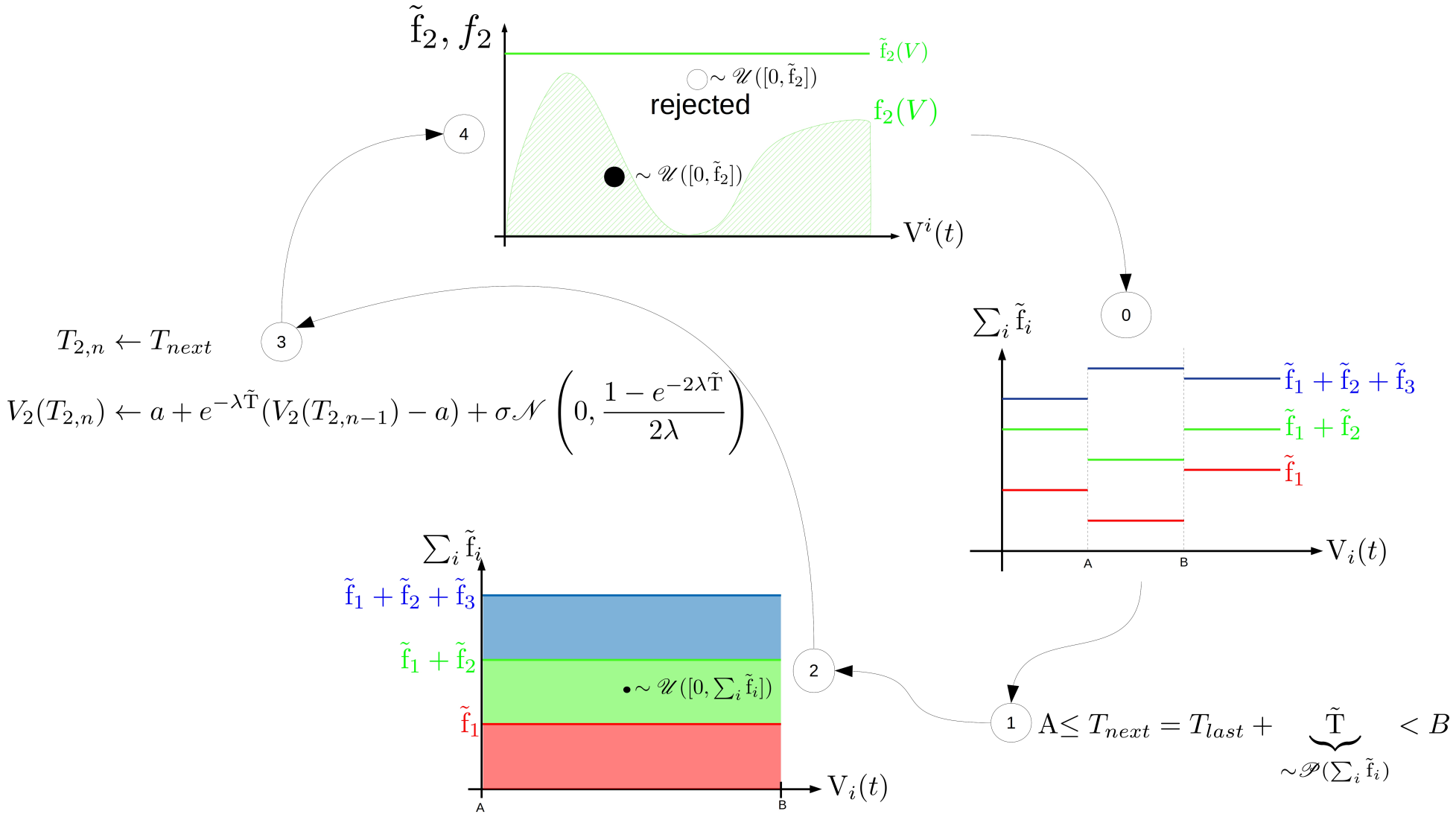


Figure 3.1: Illustration of algorithm 1

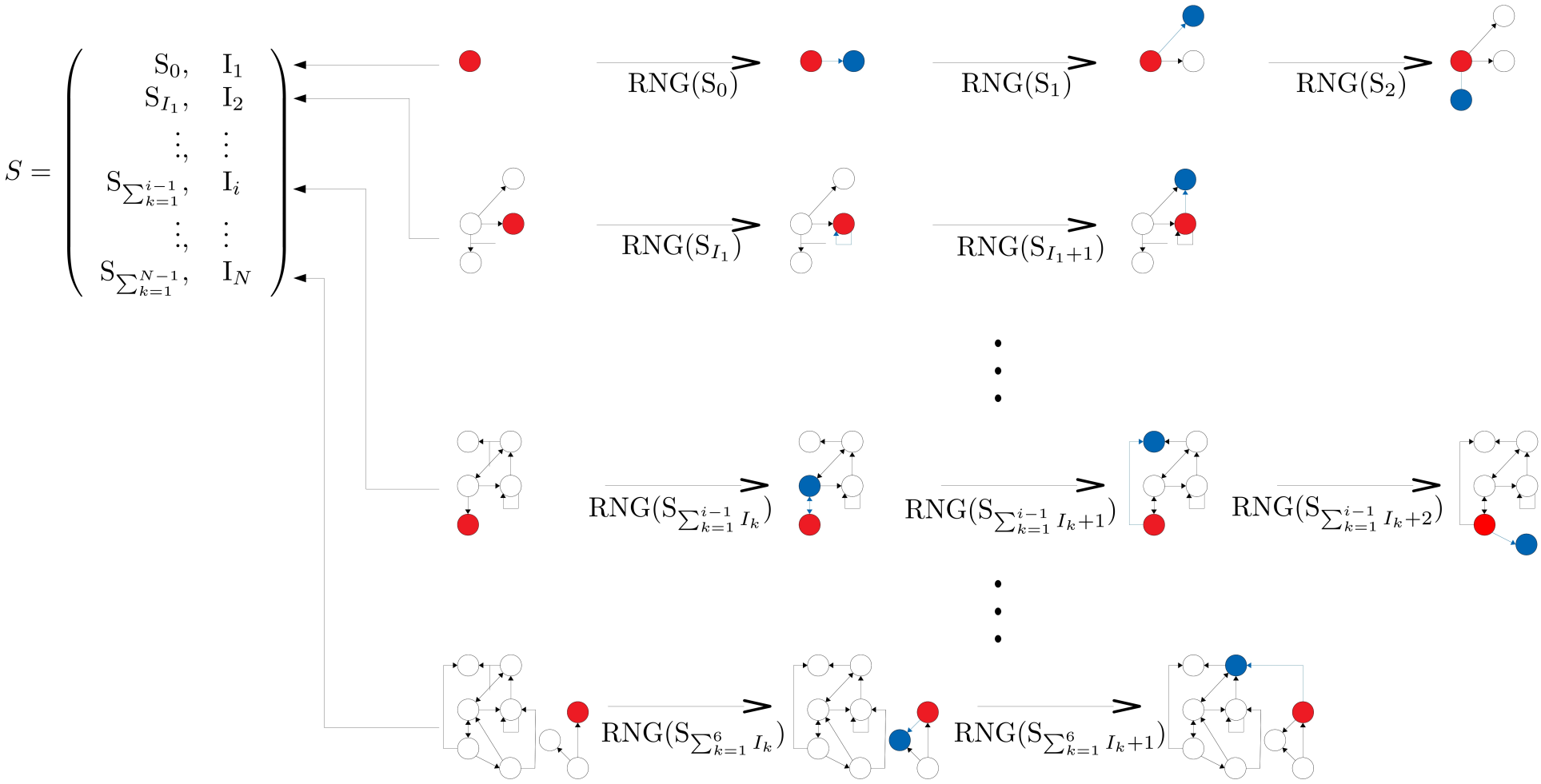


Figure 3.2: Construction of graph and vector of states

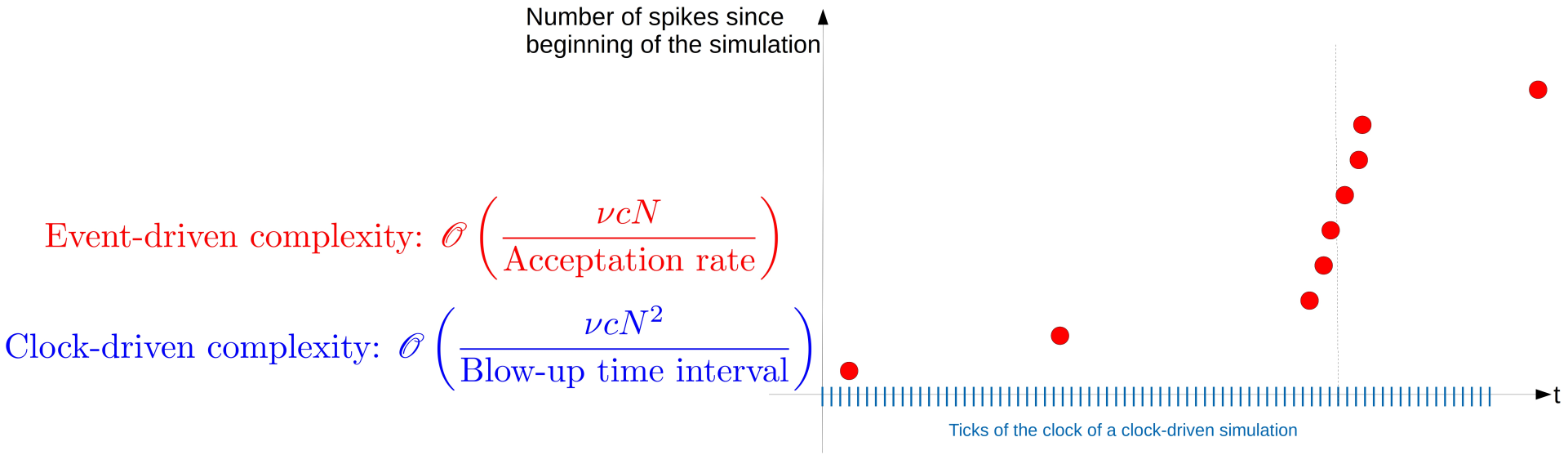


Figure 3.3: Complexity as number of events during a time interval. ν is a spiking rate, c a connectivity probability and N a number of neurons.

Chapter 4

Experiments

4.1 Performances

4.2 Experiments on the parameters of the model

Several test cases where considered, starting from the considerations on the interactions. With the notations posed above:

- variations on the constants
- $\alpha^{i,j} = \alpha$
- $\alpha^{i,j} = \alpha^i \alpha^j$, which is particularly interesting in the case where α^i or α^j is equal to zero. However, in that case, the interaction matrix is really sparse, as the graph contains a complete part and a set of independent neurons.
- $\alpha^{i,j} = \alpha^i$
- $\alpha^{i,j} = \alpha^j$
- $\alpha^{i,j}\ \mathbb{B}(p)$
- $\alpha^{i,j}\ \mathbb{B}(\frac{\ln^{\frac{1}{2},1,2}(N)}{N})$
- Variations in β, the constant of moduling the interactions, especially for the bernouillis
- clustering

On the first case, we can see there is a strong influence of the

4.3 Variations on sigma

The blow up phenomenon seems to be dependent on the amplitude of the white gaussian noise, at least for the stochastic case. Indeed, when simulating for several values of sigma, while keeping the rest of the parameters unchanged, we can see a strong correlation between the value of sigma and the apparition of the blowing up phenomenon.

4.4 Case constant interactions and variations on the probabiility of connection

As for the variations in the white noise amplitude, the interactions and the probability of connection play a great role in the blow-up. This is kind of unsurprising, as the greater the interaction the greater the chance of a postsynaptic neuron's potential to raise above the threshold.

4.5 Value of a

In the previous experiments, the drift naturally makes the potential tend to a value higher than the threshold, and so event without noise the neurons are spiking. This is kind of a natural modelling of the system: neural networks tend to be active systems, that constantly receive and transmit new information from and to the body. As we are only interested in spikes and no the exchange of information that does not directly produce a spike, it is natural to consider they naturally tend to spiking. Now the value of a is arbitrary, and it is important to test the influence of this value on the system. -¿ a = 0.9, 1.0, ..

Conclusion

This is a default conclusion, to be replaced with something less *Lorem Ipsum*.