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:

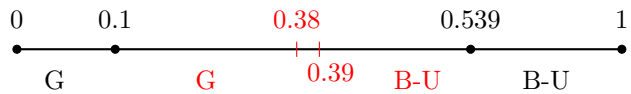
$$p \geq \inf_{\mu \geq 2\lambda} \frac{e^\mu - 1}{\mu\alpha e^{\mu x_0}} := p_c^+(x_0)$$

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.

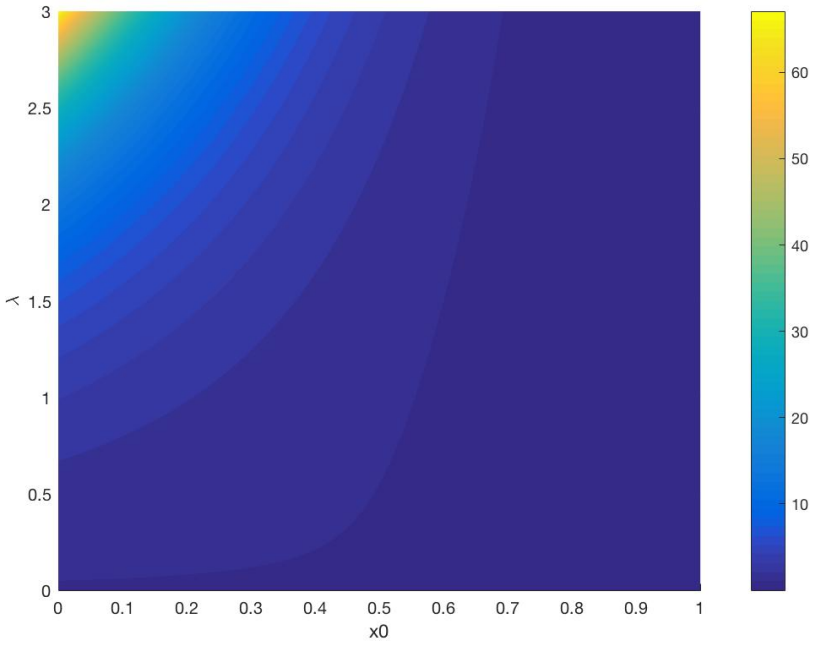


Figure 2.1: Plot of the upper bound estimate p_c^+ for the threshold p_c

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.

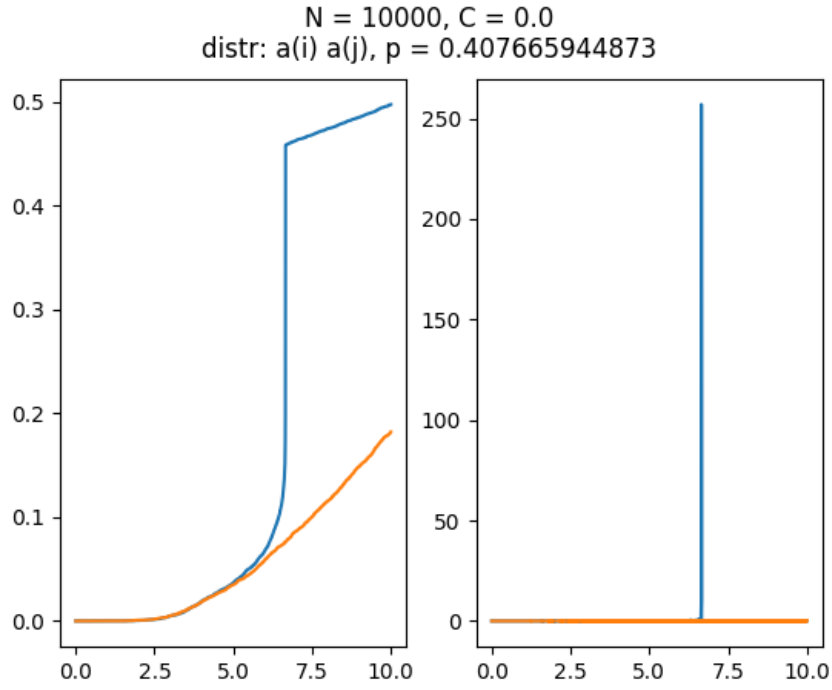
Moreover, heuristically, the system (2.12) define a a network that consist of two different component: the first is a complete sub-network, composed by neurons all connected beetwen themselves, so neurons of the subnetwork feel all the neurons of the subnetwork. The second component is made of isolated 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:

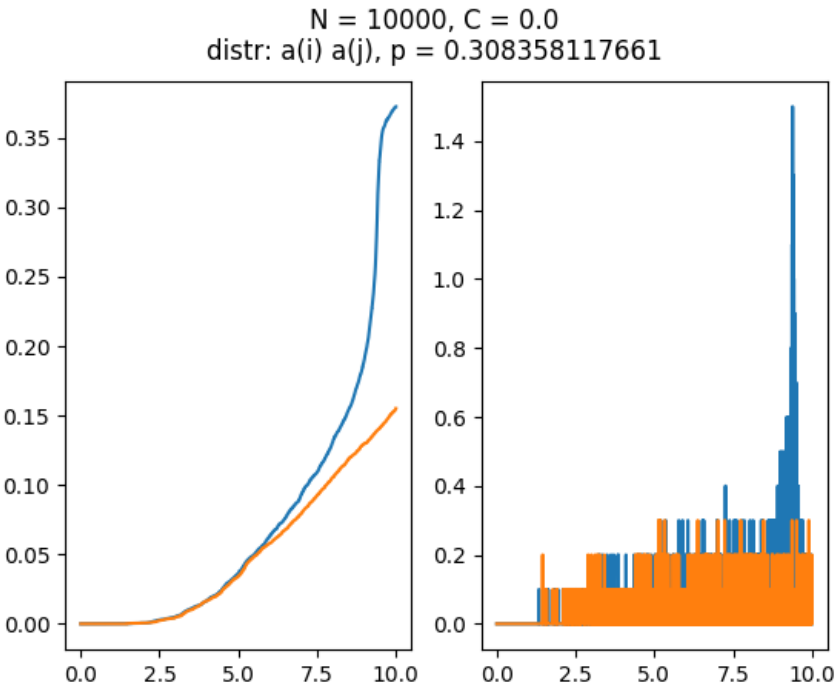
$$\alpha\mathbb{E}\Big[\alpha M_t\Big],$$

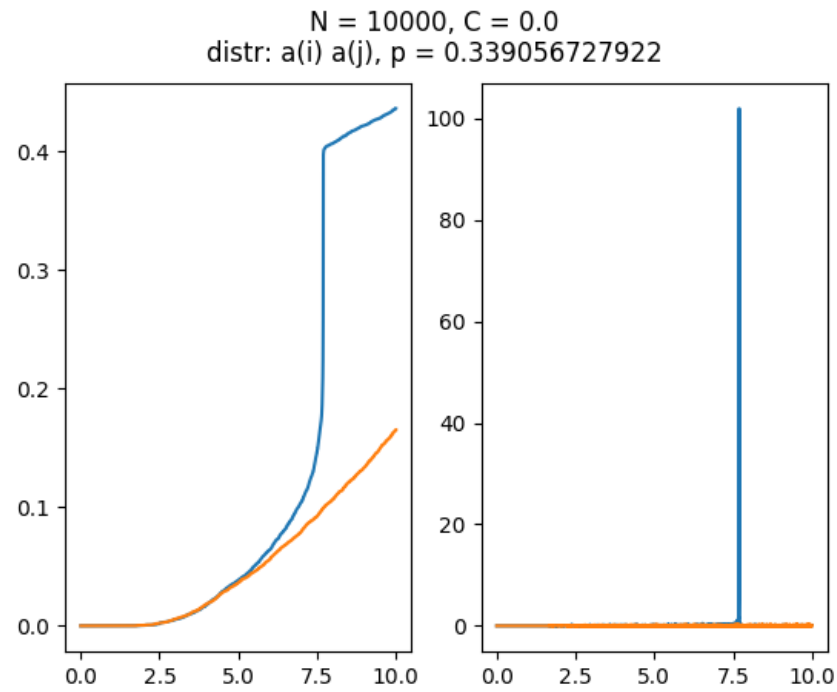
The equation (2.10) can describe the limit equation for a newtwork of neurons, that still present two different population: one of isolated neurons and the other of neurons that interact with all the other neurons, with probability p. So with interaction term:

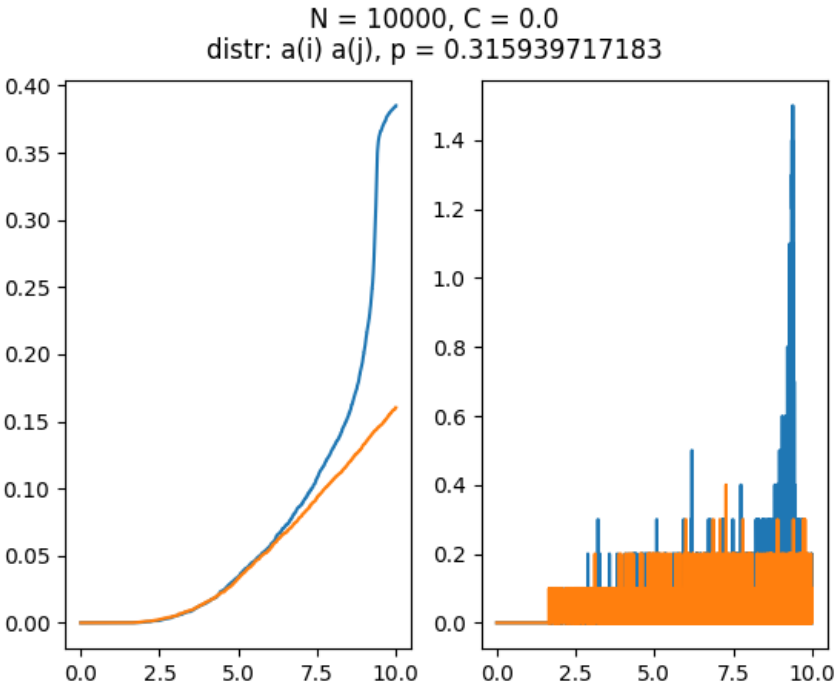
$$\frac{\alpha^i}{N}\sum_{j=1}^{N}\alpha^j M_t^j$$

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 $e(t)$. □

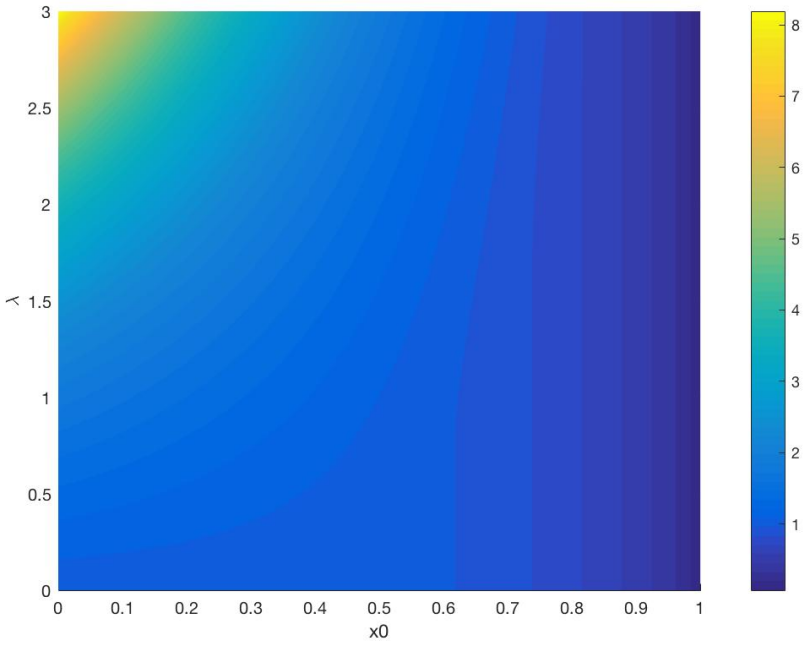


Figure 2.2: Plot of the upper bound estimate $p_c^{+,1}$ for the threshold p_c

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:

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

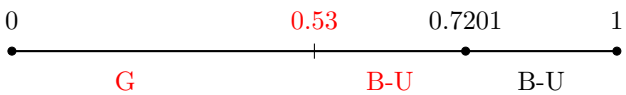
So, given a deterministic initial condition, we look for all p such that:

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

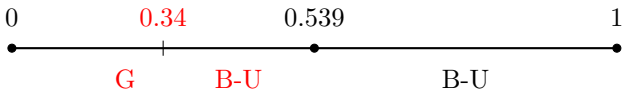
So the upper bound varying on the parameters (x_0, λ) is the same of figure 2.

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*

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

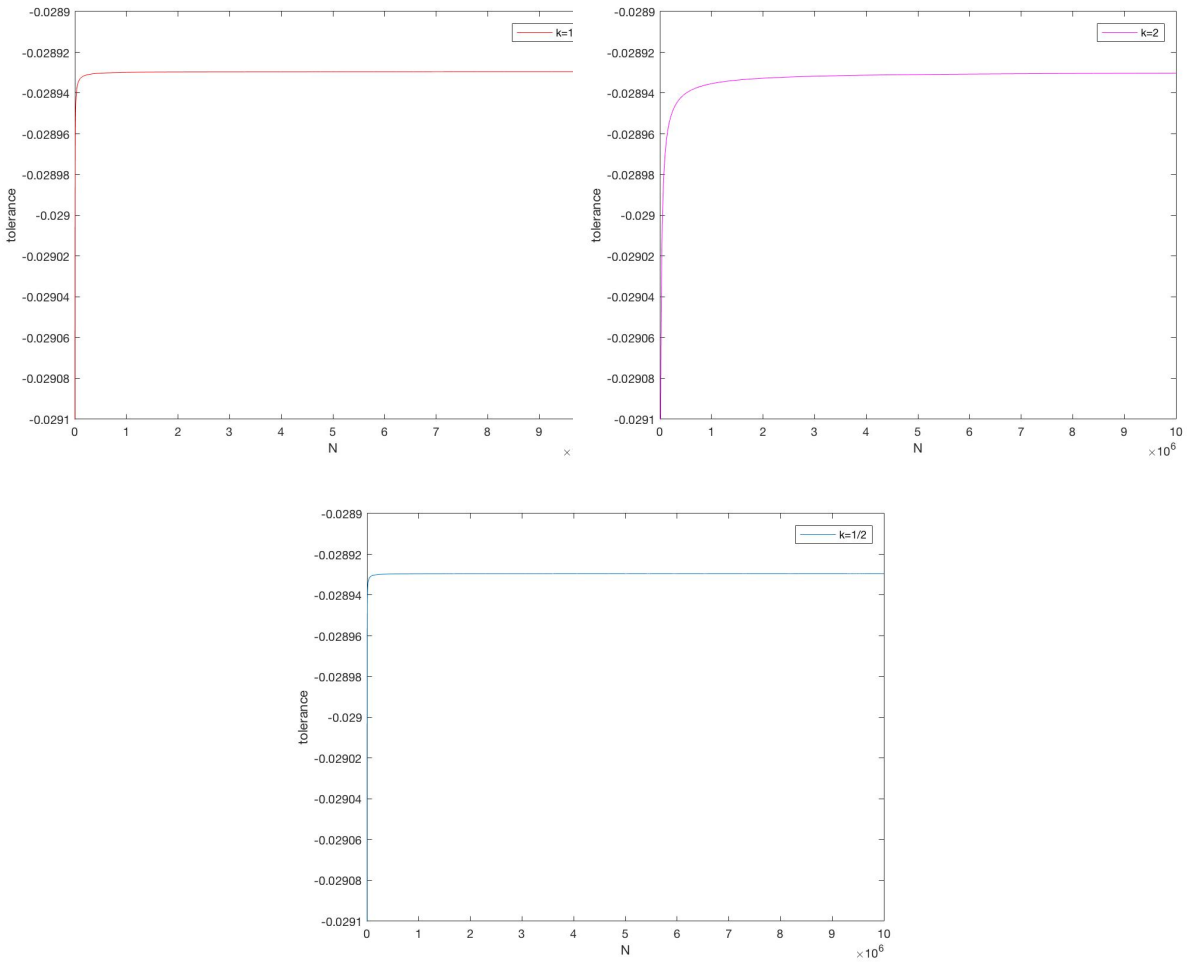


Figure 2.3: Numerical estimation of the constant $C(\beta)$ for different values of k.

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

3.1 Event-driven approach

In this chapter we aim at developping and implementing an algorithm for the simulation of a system driven by the mean-filed equation described before, and for a large number of nodes, in reasonnable times. The mean-field equation makes sense for an infinite number of actors and the actual number of neurons in most living beings is tremendous anyway (1e5 in drosophilia and 86 billions in human beings for instance).
The usual way of carrying such a numerical simulation is to compute the finite difference integration of the associated diferential equation. There are several well-known methods based on this discrete-time approach, such that Euler or Runge-Kutta, but they all suffer from the same drawbacks: namely, the computational cost which is of order N^2, meaning the number of operations necessary on order to finish the simulation is of order the square of the number of neurons.
Indeed, assuming there is on average $Nc\nu$ spikes per second (N the number of neurons, c the probability of connection between two neurons, ν the spike rate = on average a neuron emits ν spikes per second), and δt is the time step, then there is on average $\frac{cN}{\delta t}$ updates per second (at each time step, only the spiking neurons and the post-synaptic ones are updated), but the average time elapsing between two spikes is $\frac{1}{Nc\nu}$.
In order not to introduce errors due to the dynamical effect of a single spike, the time step should be so that the probability of receiving more than one spike in δt is negligible. Put it otherwise, $\delta t << \frac{1}{Nc\nu}$, and so average number of updates per second per neuron $>> N^2c^2\nu$, hence a complexity of order $N^2 perneuron$.
In order to exceed this limit, we considered a different approach. Indeed, in realistic conditions the neurons emit at low rates but with a high variability in the time intervals between successive spikes, therefore a discrete-time algorithm will spend most of its time updating the state of the neurons, while the evolution of the potential of the membrane follows a deterministic law between two successive spikes. If the dynamic of a system is driven by these spikes (or more generally any instantaneous abstract event), then a more efficient algorithm can take advantage of the determinism of the state of a neuron between two spikes to interpolate its potential whenever a spike occurs, and only update the system when a new spike is emitted.
Such an approach is called event-driven, and this is what is discussed hereafter. The complexity of this kind of algorithm is often linear in N, as a neuron influences an average of $Nc\nu$ other neurons per seconds, to be compared with the $N^2c^2\nu$ obtained before because of the very small time step necessary for not introducing a divergence too important. The simulation is also an exact solution of the equations driving the system, in that there is no approximation in the time of an event (except for the necessary rounding due to the limitation of real number representation on computers). This kind of algorithm is very effective, but much more complex to implement because of the event handling part.
Altough this approach seems promising, it is not always appliable as so. One of the necessary conditions if the existence of a deterministic path from one state to another, and in the case of systems described by stochastic equations, this hypothesis may not be fulfilled. This model is interesting because it seems to fit quite well to the assumptions necessary for event-driven simulations: the dynamic of the system is spike-driven and there is a quite deterministic evolution of the system between two spikes. On the other hand, there is a gaussian term that restrain the equation to the purely stochastic domain, as its values range in $\overline{\mathscr{R}}^+$, so the event-driven algorithm needs to be adapted to our specific needs. Thanksfully, the probability to generate a random number greater in absolute value, than an arbitrary bound decreases the higher the bound. In other words, if $X \sim (\mu, \sigma^2)$, $\mathscr{P}(\mu - r\sigma^2 \leq X \leq \mu + r\sigma^2) = 2\Phi(r) - 1$, where $\Phi : \begin{cases} \mathscr{R} \to \mathbb{R}^+ \\ x \to \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt \end{cases}$ is

r	1	2	3	4	5	6
$2-2\Phi(r)$	3.173105e-01	4.550026e-02	2.699796e-03	6.334248e-05	5.733031e-07	1.973175e-09

Table 3.1: $P(r \leq \text{X OR X} \leq -r) = 2 - 2\phi(r)$ (computed with R using *pnorm* for a centered normalised law)

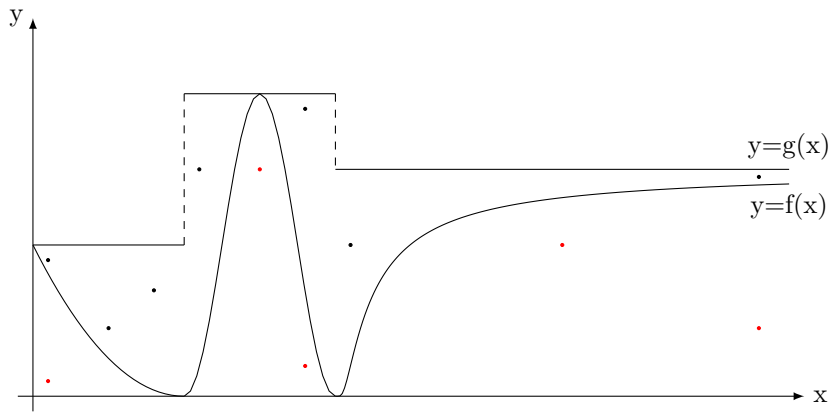


Figure 3.1: Illustration of the rejection sampling method. g(x) is the PDF of the proposed distribution Y and f(x) is the PDF of the target distribution X. The points are first generated following the Y distribution, then sampled and accepted only if they follow the X distribution, i.e. their value is less than the one of the pdf f. There are five accepted points (in red) and seven rejected points.

the cumulative distribution function of the gaussian distribution. Here r can be viewed as a confidence indicator, and so we can define a bound that will be almost surely an upper bound of the white noise.
The first question that arise from this remark is how good is this confidence interval? With an infinitely large value of r the confidence is nearly 100%, but then the upper-bound of the white noise is very large. On the table 3.1 the actual values of the probability to generate out-of-boundary values are displayed. Even small values of r give a high enough confidence for several simulations, depending on the parameters chosen for the simulation, particularly the number of neurons. This brings back the discussion about the complexity of the algorithm:

Add exact complexit

3.2 Rejection sampling method

As a reminder, here is the model of the system as presented in the previous chapters:

Equation of the model

This kind of stochastic equation is often solved using a rejection sampling method. This method is used for generating random numbers from a target distribution X with the probability distribution function f(x), using another distribution Y of probability distribution function g(x). The idea is that generating numbers from the proposed distribution Y shall be easier than generating numbers directly from X, so that one can generate a large set of numbers from Y and then sample them, accepting with a probability $\frac{f(x)}{g(x)}$. There is a condition of course that f and g must satisfy $f(x) \leq g(x)$ for all x. Also, even if that is not necessary, there is another condition that $f(x) \approx g(x)$, otherwise the rejection sampling method rejects too many propositions. The figure 3.1 exposes this process, with a non usual pdf f and a very classical piecewise constant pdf g. As can be seen on the figure 3.1 the acceptation rate is often quite low because it can be hard to find a pdf g matching the target pdf f. Also the approximation pdf must at any time have higher values than the target pdf, meaning sometimes it does not even exist.

Put the equation of the mo

3.3 Definitions

With this approach some definitions given before are no longer appliable to the system.
The threshold is then defined as:
At any given time, any neuron i of potential V_i has a chance of firing which is function of

$$f(V_i) = S * (Pos(V_i - V_T))^E, \text{ where } Pos(x) = \begin{cases} x & \text{if } x > 0 \\ V_R & else \end{cases}$$

V_T is chosen equal to one, and V_R is equal to zero for all neurons (for simplicity purposes), but they could be chosen more randomly. A threshold value too low (too close to the reset value) is problematic, as it means that potentials are going to reach their threshold quickly after a neuron has spiked (in which case the system is "stuck" in a perpetual series of spikes). By design only one neuron can fire at a given time (the spike trains are point processes), and if a cascade is cannot normally happen (except in cases where some values are rounded to zero due to the limitations in floating point precision, but more on that later) that does not solve the issue of blow-up, as a similar phenomenon can still occur (the rate will not tend to infinity, still it tends to very high values). The blow-up is indeed newly defined as (Pertinence de comparer à N lorsque tous les neurones ne spikent pas ? Pourquoi pas plutôt quelque chose du type $1/\delta_N -> \infty$, avec δ_N l'intervalle entre deux spikes ?):

$$\frac{1}{\delta_N} > Constant, \text{ typically N}$$

where δ_N is an interval between two spikes (or a mean interval, or a moving average). The constants S and E are chosen so that the chances of fire are high even for values of the potential low above the threshold (typically, $S = 10^5$ and $E = 5$, while values of potentials at spiking times are around [CALCULER VALEUR PRECISE, DE MEMOIRE 1.4]).

The system is driven by a stochastic PDE and discontinuities are randomly introduced depending on the value of parts of the system. There are two common ways of simulating a stochastic PDE. The Euler-Maruyama method for numerical approximation of stochastic differential equations results in the construction of a Markov chain on the interval $[0, T]$ of simulation. While quite simple to set up and understand, it is wastefull for processes that spend a lot of time not changing, in this case not spiking. In addition, the resolution of the time interval shall be low enough to capture the meaningful variations in the system, here the moments of spikes and the blowing-up behaviour, as well as to keep the simulation precise enough. Spikes are diracs, discontinuities of the system, and the blow up happens when the firing rate tends to infinity, or to spell it otherwise, when the time interval between spikes tends to zero.
The second method is the rejection sampling, which is a common method for generating random numbers from distribution of not well defined cumulative distribution function http://www.aip.de/groups/soe/local/numres/bookcpdf/c7-3.pdf. Here the stochastic process is the network of neurons, and its intensity directly depends on the potentials of the neurons, following the function $f : \mathbb{R} \to \mathbb{R}^+$ described above. In this method two numbers (for the one dimension case) are generated, $(t, u) \in \mathbb{R} + \times [0, 1]$. They represent a point in the plane and depending on whether the point can be placed under the curve of the desired cumulative distribution function or not the point will be declared accepted or rejected. In order to achieve this in practice the value of $\frac{f(x)}{f_{max}(x)}$ is compared to a number uniformly generated between zero and one. The approximation function is used in order to increase the probability of generating a point under the curve (on the full spectrum of real positive numbers, the chances are null).

INCLUDE GRAPHI EXAM- PLE OF THE PRO CEDURE ALIKE THE ONE DRAWN ON BOAR

The f_{max} here can be computed by a simplified, limit in time version of the model. Indeed, a rough estimate can be created by dividing the equations in sums of which the maximum value in time will be taken.

$$f_{max} = \max_t(a + \exp^{-\lambda t}(y_t^i - y_S^i)) + \max_t(\sigma \mathbb{N}(0, \frac{1 - \exp^{-2\lambda t}}{2\lambda}))$$

The maximum proposed here is not correct. Indeed, there are no boundaries for a normally distributed random number, hence the white noise cannot be bounded and the variations for the potentials are in $\mathbb{R} \cup \{-\infty, \infty\}$... But! The chance of generating a number of absolute value larger than the variance decreases the farther away this number is from the variance. Thus it is possible to consider, with a certain degree of confidence, that no numbers will be generated outside of a predefined interval, the bounds of which will constitute our maximum. More precisely, given a random variable X following a normal law X $\mathbb{N}(\mu, \sigma^2)$, then it is possible to compute $P(\mu + k\sigma \leq X)$. The table 3.1 gives values of the chances of generating a number outside the interval $[\mu - k\sigma, \mu + k\sigma]$.
In the current implementation k has been fixed at 5, as the number of neurons will hardly be higher than 10^5 for memory reasons (the graph of interaction takes $p \times N^2 \times 64$ bits of ram memory, p being the probability of connection between two neurons in a Erdos-Renyi graph, 64 being the number of bits necessary for representing an integer on a typical computer nowadays and for $N = 10^5$ this value is higher than the typical amount of ram available on a computer, even for dedicated comptuting machines). There is also an influence of the number of spikes (accepted or rejected) generated during the simulation: each time the simulator decides whether the spike is accepted or not the true value of the potential is computed therefore the value of the noise is computed and there is a chance of generating a normally distributed number higher than the bound we have set.

So tosummarize, we have chosen to use a thinning procedure to compute the solution of the stochastic partial differential equation described in the **??**. The thinning procedure necessitates the use of an approximation function, which must be in any point higher than the function to simulate/compute. Even though a part of our model is a gaussian white noise, which is unbounded, it is still possible to use a pseudo-boundary for the noise, that is to say a number that will constitute an upper-limit with a certain degree of confidence. The limit directly depends on the confidence interval we want for our simulation, so this solution is both easy and flexible. Computing this pseudo-boundary necessitates only values that are necessary to compute the value of the gaussian white noise, so this solution does not increase the amount of computations needed. Actually it is much less expensive, in processor time, to compute the value of the upper-limit, as the actual value of the variance (which relies on an exponential in our case) is not needed (a maximum in time of the variance is enough, and adds in the confidence). Also, it is important to remark that even if we have focused on giving an upper-bound to the noise, there are other parts in our model that are approximated too, and so values of the noise not too much above the upper-limit may not make our approximation function f_{max} lesser than the actual potential, so the solution is robust even to small errors.

Finally, we can remark that we are sampling the noise anyway, meaning there is no way for us to know that even if the noise at the time of spike is indeed in the boundaries, it has not crossed the boundaries during the time inbetween two consecutive spikes. So what if there are errors in the evaluation of the noise? Two cases emerge. First, there is a bad approximation of the noise of the spiking neuron. So this neuron is certainly the good one to pick for a spike but it may have spiked a bit earlier if we had the information of the real value of its potential. The second case is more criticial, as this is the case where a neuron has been chosen for spiking, but the potential of one of the non chosen neurons is higher than expected, meaning it had actually more chance to spike. Of course the neuron we have chosen may actually still be the good one, then the error does not matter at all, the next computation of the potential shall rectify the mistake. Yet, even if we imagine that having another estimation for the potential would have changed the spiking neuron, given that we are dealing we at minimum 10,000 neurons, and more probably 100,000 neurons, and generating around the same number of spikes, having even a handfull of mistakes would not mean a large deviation of the final result (if a neuron should have spiked but does not, and this because of the noise, not the configuration of the network or the intensity of the spikes or whatever, then it will spike just a bit later, and excepting in the case where we speak about very critical neurons of very specific networks, this kind of mistakes can be considered as a good tradeof for speed).

At this stage we have a first draft for the thinning procedure algorithm, but we can improve it a lot. For now we have focused on improving the efficiency of and discussing about the quality of the approximation function f_{max}, but f is not the real equation simulated here, it just helps defining a probability for a neuron to spike, depending on the value of its potential. But as the potential evolves in time, the approximation is always false, and by a bigger margin at the beginning of time than at infinity, since the approximation function is built using the values for $t \to \infty$. Unfortunately, the values of t (the x-axis part of the points generated for the thinning method) have a lot more chance to be small as they are generated using an exponential distribution (we are dealing with poisson processes so the spikes must be exponentially distributed in time [enfin ici du moins en l'occurrence c'est en temps]). This is also an issue for the random generation, as pseudo-random number generators cycle and generating more unuseful numbers may lead the generator to cycle.

A common solution to avoid this is to bound the research of accepted points in time: we define a time interval of research and then look for points in this interval. Once a certain condition is met (for instance here every time we have an accepted point or when the time for the next potential spike is greater than the time limit) we stop or change the time interval. This way the approximation function is way closer to the real function and we can drastically improve the amount of accepted points.

INSERT HERE GRAH O THINNIN METHOI WITH TIME IN- TERVALS

3.4 Algorithm in pseudo code

Finally here is, in pseudo code, the algorithm used for simulating the equation driving our model.

3.5 Improving efficiency

This section will mainly focus on speed and memory usage. The algorithm presented above works perfectly except that the computations involved here can be very long in processor time, and that the amount of

PENSER METTRE UN PEAC SUR OPENMI LES AL- TERNA-

Algorithm 1 Pseudo code of the thinning algorithm used for simulating the system

Initialize all parameters
repeat
  repeat
    determine an interval $[t_{n-1}, t_n]$ on which sampling
    compute the array of $f_{max}(Y_{t_{n-1}})$ and $\sum f_{max}(Y_{t_{n-1}})$ on interval $[t_{n-1}, t_n]$
    $t_n \sim t_{n-1} + \mathscr{E}(\sum f_{max})$
  until not in good interval or all f_{max} are at 0 $u \sim \mathbb{U}([0,1]) \rightarrow \mathbb{P}(\text{spiking neuron is neuron i}) = \frac{f^i_{max}(Y^i_{t_n})}{\sum_j^N f^j_{max}(Y^j_{t_n})}$ $u \sim \mathbb{U}([0,1]) \rightarrow \mathbb{P}(\text{accepting spike of neuron i}) = \frac{f^i(Y^i_{t_n})}{f^i_{max}(Y^i_{t_n})}$
  if spike is accepted **then** update the potentials of all postsynaptic neurons
until do not have the good amount of accepted spikes

memory for running it is potentially tremendous. On the speed part, the algorithm has been implemented in C and optimised using -O3 in order to take advantage of the gain of speed of the low level languages. Dedicated libraries (here openmp) can and have been used in order to improve the speed of some parts of the simulation, in the parts where a state variable needs to be changed for all or a large amount of neurons. Some specific interaction cases, like full connection (including selfconnections), complete interaction graph and independence (no connections) are encoded so that the connection graph is not needed, improving speed and memory consumption. but on the generic case, even though the full matrix of interactions is not stored (only the relevant parts), the memory for storing the interaction graph takes around $p * N^2$, where p is the probability of interaction and N the number of neurons. For even 10^5 neurons that makes about 1 GB, given 1 byte is enough to store the index of neurons (and the addresses in memory)(spoiler: it is not, and takes rather 4 times more) and with a probability of connction of 0.1... where the probability of connection was 1 in the mathematical papers.

Still, when the interaction graph is too big but there is no way to skip the generation of the interaction graph, there is a way to save a lot of memory by storing seeds instead of a boolean about whether there is an interaction or not. When generating an interaction graph with a random connection probability, for each neuron, instead of storing an array containing alll the postsynaptic neurons, one can rather store the state of the random number generator (for instance, in Mersenne Twister, it is about 128 bits, and it is one of the largest state for a classical random number generator) just before randomly chosing the postsynaptic neurons. This way, each time a neuron is spiking, one can regenerate the same graph, and more interestingly the specific part of the interaction graph of interest (that is to say only the postsynaptic neurons of the spiking neurons, and no the entire graph of interaction). While a lot slower, this method is probably one of the most efficient in memory, taking advantage of the pseudo character of the random number generation in computer science.

This method is also parallizable with openmp or any library alike it, but it is more difficult to achieve if one want to guarrantee the reproducibility of the results. The parallelization is possible because if one cuts the array of the postsynaptic interactions in k parts, then k random number generators can be used for generating the interaction graph. The issue here is on the initialization of the random number generators, as we want to avoid the case where there is an overlapping in the sequences in use by two threads at the same time. A RNG like the Mersenne Twister, with a very large pseudo period is great for this purpose, as it allows to cut the pseudo period so that two processes correctly seeded will never overlap. This considerations are also needed if one wants to run several simulations in parallel. [METTRE PUBLI BENNIE]

3.6 Implementation

As said before, the algorithm was finally implemented in C (and prototyped in Java). There are several reasons for that. The C language is close enough to the machine language to be one of the fastest possible and allows a precise management of the memory usage. Most compiler (as gcc, the one we used) also have options for improving the speed and memory consumption of the program. These options are also supported by the version of the Mersenne-Twister random number generator used in the implementation. There are two of them, on for integer generation and one for double precision generation, and they can be found at http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html. O this site there are more information about Mersenne-Twister generators in general, as this page has been created and is maintained by one of their creator, Makoto Matsumoto.

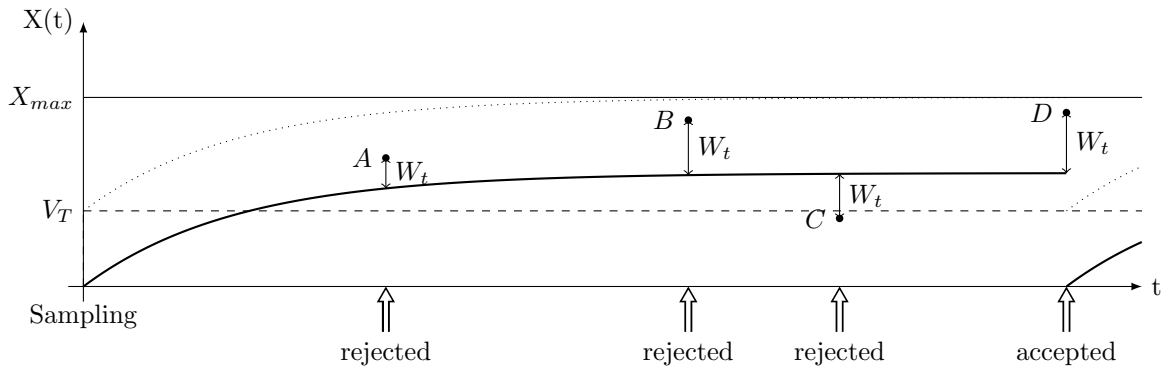


Figure 3.2

The computations at stake here involve potentially both very low and very high values. For instance the time of spikes are exponentially distributed. They are obtained using an exponential distribution of parameter the sum of all the approximation functions (on the current interval of work). This value can typically raise up to 10^5 or 10^6 (consider the same number of neurons, their potential close to the threshold).

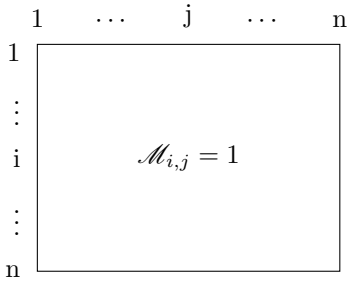
For this reason some

3.7 A word about the rng

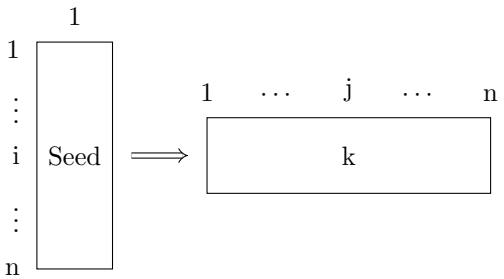
Mersenne twister, and not xorshift because the mersenne allows for a reproduction of the simulations, which is also necessary for a part of the algorithm. It is well known, and has a huge period which is important for a part of the algorithm. Also there are a lot of good implementations, some of the creator itself who actively update them when bugs or improvements are found. The method also allow for good generation of random real numbers (not only integers)

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. Of course the maximum value M must be quite high, compared to $U + 1$ in order for this method to be effective. For instance, if $U + 1 =$, there is a 49% of rejection. If the rest of the euclidean division of M bu $U + 1$ is U, the



(a) Matrix of children $\mathscr{M}_{i,j} = 1$ if j is children of i and 0 otherwise.



(b) Vector of seeds and reconstructed vector of children

```
1: RNG: a random number generator
2: p: probability of connection between two neu-
   rons
3: function GENERATE INTERACTION MA-
   TRIX(RNG, p)
4:     for all (i, j) ∈ {1, ··· , n}^2 do
5:         M_{i,j} ← RNG.B(p)
```

(c) Generation of a matrix of children

```
1: RNG: a random number generator
2: p: probability of connection between two neu-
   rons
3: V^s: vector of states
4: V_i^c: vector of children of neuron i
5: function GENERATE RECONSTRUCTIBLE IN-
   TERACTION MATRIX(RNG, p)
6:     for i ← 1 to n do
7:         V_i^s ←RNG.STATE
8:         for j ← 1 to n do
9:             RNG.B(p) ▷ Generate, without storing them, the
10:    Some code ···
11: function RECONSTRUCT CHILDREN OF(i)
12:    for j ← 1 to n do
13:        j is children of i with probability p
```

(d) Generation of a vector of rng states

Figure 3.3: Representing the two algorithm for generating an erdos-renyii graph

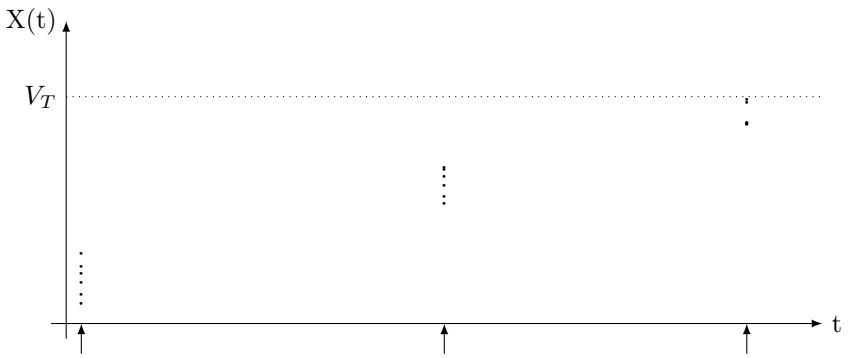


Figure 3.4: Illustration of the initialisation process

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*.