UNIVERSITY OF CORDOBA
INSTITUTE OF POSTGRADUATE STUDIES

UNIVERSITY MASTER'S DEGREE

PHYSICAL TECHNOLOGY: RESEARCH AND APPLICATIONS

MASTER'S THESIS

Exploring the relationship between Hawkes processes and self-organized criticality in living systems

Antonio Rivas Blanco

Tutor(s): Jorge Hidalgo Aguilera and Serena di Santo

Line of research: Modelling of complex systems and their interdisciplinary applications

Córdoba, 07/2024

Autorización de defensa del Trabajo Fin de Máster

Tutor 1: Jorge Hidalgo Aguilera
Tutor 2: Serena di Santo

INFORMAN: Que el presente trabajo titulado *Exploring the relationship between Hawkes processes and self-organized criticality in living systems* que constituye la memoria del Trabajo Fin de Máster ha sido realizado por Antonio Rivas Blanco y AUTORIZA/N su presentación para que pueda ser defendido en la convocatoria 1ª con fecha 23/07/2024.

Firmado en Córdoba a X de julio de 2024

Tutor 1: Jorge Hidalgo Aguilera

Tutor 2: Serena di Santo

Declaración de autoría

Nombre y apellidos: Antonio Rivas Blanco
DNI: 49832223D

DECLARA

1. Que el trabajo que presenta es totalmente original y que se hace responsable de todo su contenido.
2. Que no ha sido publicado no total ni parcialmente.
3. Que todos los aportes de otros autores/as han sido debidamente referenciados.
4. Que no ha incurrido en fraude científico, plagio o vicios de autoría.
5. Que, en caso de no cumplir los requerimientos anteriores, aceptará las medidas disciplinarias sancionadoras que correspondan.

Firmado en Córdoba a, X de julio de 2024

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Chapter 1

Introduction

ESTRUCTURA:

The following Master's thesis is divided into 5 chapters. In the first chapter, we will introduce the concept of criticality in complex systems, both in living and non-living systems. We will also make a brief summary of stochastic and their relation with point processes and Hawkes processes, which are the main focus of this work. After that, we will present the objectives of this work. In the third chapter, we will present the methodology used in this work, including the simulation of Hawkes processes, the phase diagram of the process, and the avalanche analysis. In the penultimate chapter, we will present the results of the simulations, including the phase diagram of the process, the susceptibility analysis, and the avalanche analysis and we will discuss them. Finally, in the fifth chapter, we will present the conclusions of this work and the future lines of research.

1.1 Criticality and its presence in living systems

HABLAR AQUÍ DE LOS EXPONENTES τ, α

For the first simulations, we will fix $n = 1$, making our self-excitation critical.

The concept of criticality is a central idea in the study of complex systems. It is a state of the system where the dynamics of the system are scale-invariant, which means that the dynamics of the system are the same at different scales. This means that the system is at the edge of chaos, where the system is neither ordered nor disordered. This state is characterized by the presence of avalanches of activity that follow a power-law distribution. This means that the probability of having an avalanche of size s is proportional to $s^{-\tau}$, where τ is the exponent of the power-law distribution. This state is present in many systems such as earthquakes, forest fires, or the brain. In the brain, it is believed that the critical state is the optimal state for information processing, as it allows the system to respond quickly to stimuli and to have a high capacity for information storage. This state is also associated with the presence of avalanches of activity that follow a power-law distribution. This means that the probability of having an avalanche of size s is proportional to $s^{-\tau}$, where τ is the exponent of the power-law distribution. This state is present in many systems such as earthquakes, forest fires, or the brain. In the brain, it is believed that the critical state is the optimal state for information processing, as it allows the system to respond quickly to stimuli and to have a high capacity for information storage.

1.2 Point processes

Within the large framework of complex systems, stochastic processes lend us a hand to decypher properties of living systems, bridging randomness with structured behaviour. This processes are used to model the dynamics of systems which evolve randomly in time. This is why they are ideal for describing natural phenomena such as the spread of diseases [1], social networks [2] or ecological systems [3]. Mathematically, a stochastic process is a collection of random variables [4], generally ordered in time $\{X_t\}_{t\in T}$, where t is the time and X_t is the system state at time t. T is the time index set, which can be discrete or continuous, in this work we will focus on the discrete case because we are interested in the study of point (Hawkes) processes for modeling neurons.

Point processes are a type of stochastic process that describe the occurrence of events in time or space. We will be interested in time point processes because we are going to model the spiking activity of neurons. For our purposes, they will be characterized by two parameters, the time of occurrence of the events t_k and the intensity or rate of occurrence of these events λ. This rate tell us how likely is that an event occurs at time t given the history of the process (probability density function, PDF) as pictured in Figure 1.1.

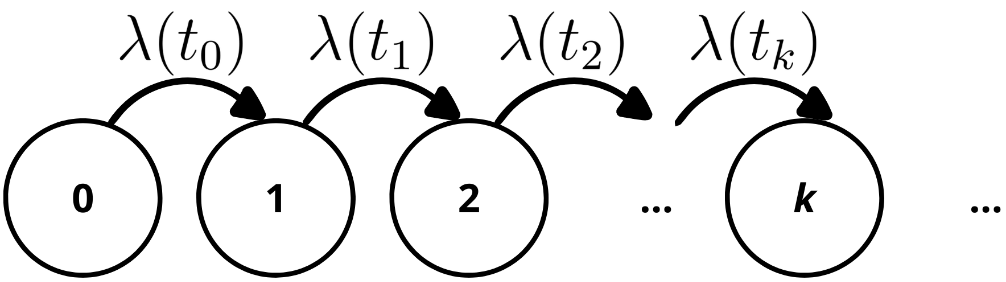


Figure 1.1. Representation of a point process. The intensity function $\lambda(t)$ is a time-dependent function.

1.2.1 Poisson processes

In general, the rate is a function of the history of the process, which makes the process non-Markovian, but in our case, it will be a Markovian process, which means that the rate depends only on the last event that occurred as we will see. An example of a Markovian point process is the Poisson process, which is a simple and one of the most studied point processes because they are present in many everyday situations such as the arrival of customers at a store, occurrence of defects on a Production line. They are also present in some physics phenomena, for instance, the decay of radioactive particles or the arrival of photons at a detector. These processes are characterized by a rate of occurrence of events λ. The dynamics of these processes are described by the Poisson distribution which is the probability distribution of a random variable N such that the probability that $N = n$ is:

$$P(N = n) = \frac{\lambda^n}{n!}e^{-\lambda}. \tag{1.1}$$

Furthermore, the mean value and the variance of the distribution are also equal to λ. Poisson processes can be homogeneous or inhomogeneous, depending on whether the rate is constant or time-dependent.

In Figure 1.2 we can see an example of a homogeneous Poisson process.

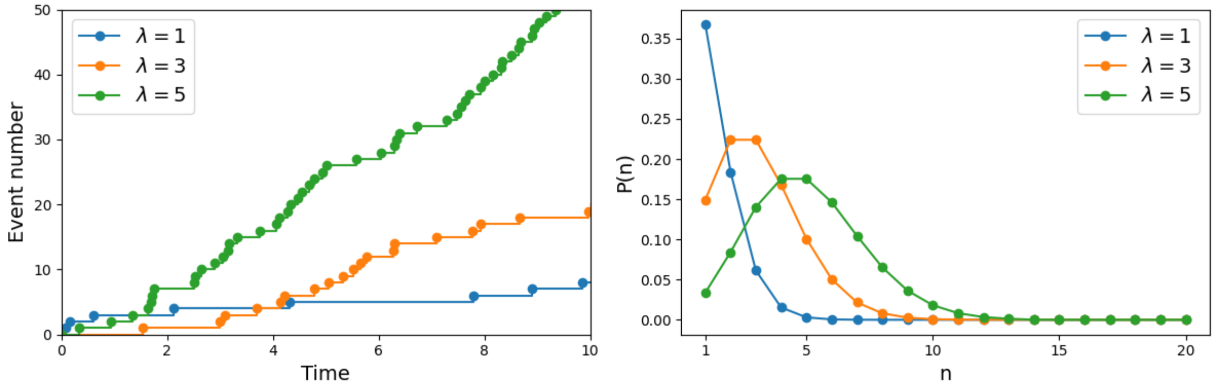


Figure 1.2. Left: event number in time for different rates. Right: Probability of having a certain number of events for different rates.

1.2.2 Hawkes processes

On the other hand if we consider a non-homogeneous Poisson process, the rate is a function of time, $\lambda(t)$, which is the case of the Hawkes process. The rate can be written in several ways [5, 6, 7, 8]. We will use the expression from [5]:

$$\lambda(t|t_1, \dots, t_k) = \mu + n \sum_{i=1}^{k} \phi(t - t_i), \tag{1.2}$$

where μ is the background rate of a homogeneous Poisson process, n is a parameter that controls the strength the self-excitation, and $\phi(t)$ is the kernel function that describes the influence of the past events on the rate of occurrence of the events. The kernel function is a non-negative and monotonically non-increasing function that integrates to 1. Typical choices for the kernel function are the exponential or the power-law functions. In this work we will focus on the exponential kernel. From Eq 1.2 we can see that the rate depends on the history of the process, making it non-Markovian in general, but with an exponential kernel, the process becomes Markovian. The kernel function can be written as: $\phi(t) = \sum_{t_i<t} \alpha e^{-\beta(t-t_i)}$ so the rate becomes:

$$\begin{aligned} \lambda(t) &= \mu + \sum_{t_i<t} \alpha e^{-\beta(t-t_i)} \\ &= \mu + \sum_{\underbrace{t_i < t_k}_{t_k\text{: last event}}} \alpha e^{-\beta(t-t_k+t_k-t_i)} \\ &= \mu + e^{-\beta(t-t_k)} \underbrace{\sum_{t_i<t_k} \alpha e^{-\beta(t_k-t_i)}}_{\lambda(t_k)} \\ &= \mu + e^{-\beta(t-t_k)} \left(\lambda(t_k) - \mu + \alpha\right). \end{aligned} \tag{1.3}$$

Where we have used the following expression for the rate of the Hawkes process at time t_k:

$$\lambda(t_k) = \mu + \sum_{t_i<t_k} \alpha e^{-\beta(t_k-t_i)} \Rightarrow \sum_{t_i<t_k} \alpha e^{-\beta(t_k-t_i)} = \lambda(t_k) - \mu + \alpha \tag{1.4}$$

Despite being a Markovian process, it is still an inhomogeneous Poisson process because the rate is not constant. In addition, it is a self-exciting process, which means that the occurrence of an event increases the probability of the occurrence of another event. This is why it is used to model the spiking activity of neurons, where the occurrence of a spike increases the probability of the occurrence of another spike. This self-excitation will enable the appearance of bursts of activity that we will measure. The parameters chosen for the kernel function will be $\alpha = \beta = 1$ and we will vary the background rate μ from values much smaller than 1 to values greater than 1. In Figures 1.3 and 1.4 we can see typical diagrams of Hawkes processes with these parameters.

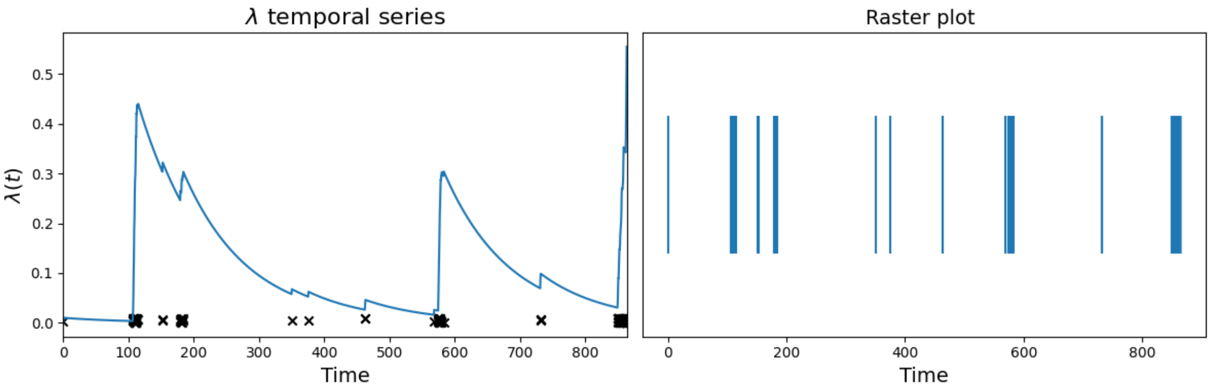


Figure 1.3. On the left, a temporal series of $K = 150$ events of a Hawkes process with $\mu = 0.01$, on the right, a raster plot of the same process.

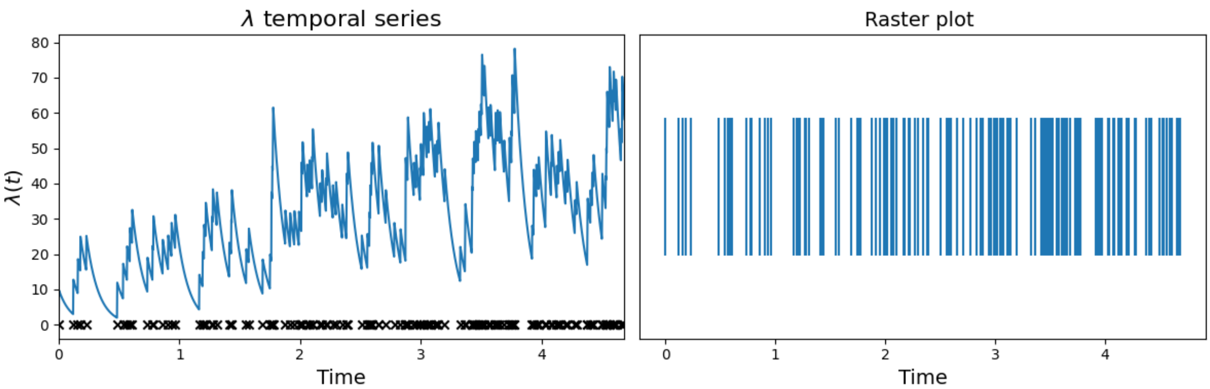


Figure 1.4. On the left, a temporal series of $K = 150$ events of a Hawkes process with $\mu = 0.01$, on the right, a raster plot of the same process.

As shown in Figure 1.3, when the background rate is smaller than 1, events are less likely to occur, but when they do, they tend to form avalanches of activity thanks to the self-excitation. On the other hand, when the background rate is greater than 1, events occur more frequently, forming avalanches of activity more frequently and longer, as shown in Figure 1.4. If we ignore the time of occurrence of the events and we focus only on the structure of λ and therefore of the events, we can see that the process with $\mu = 0.01$ has a bursty structure, while the process with $\mu = 10$ has a more regular structure. This phenomenon is exposed in Figures 1.5 and 1.6.

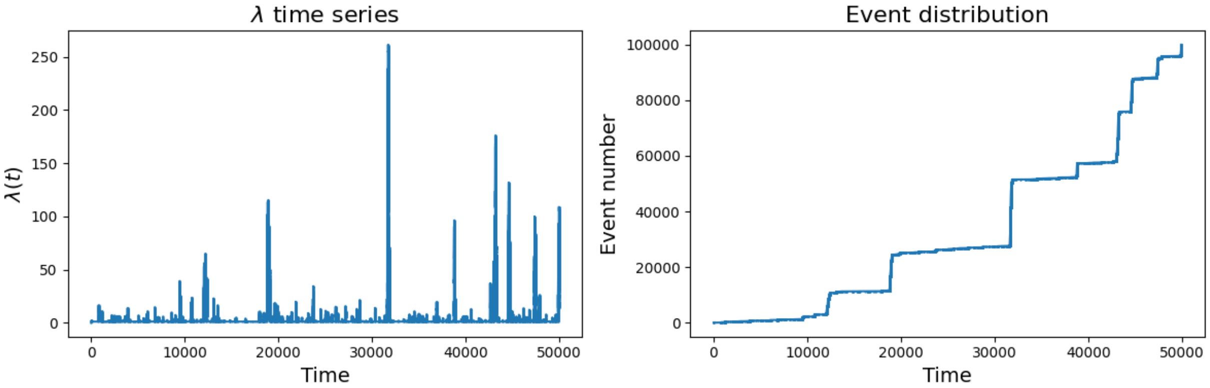


Figure 1.5. First, a temporal series of $K = 10^5$ events of a Hawkes process with $\mu = 0.01$, on the right, the event distribution.

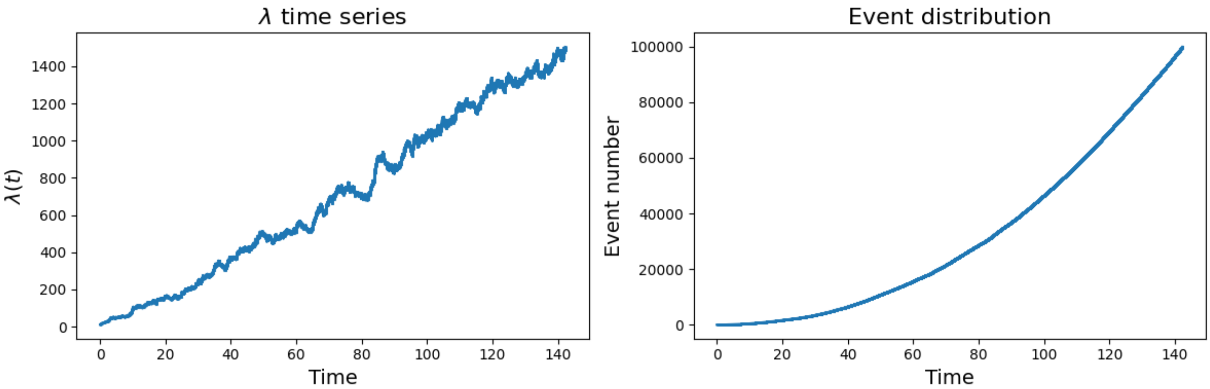


Figure 1.6. First, a temporal series of $K = 10^5$ events of a Hawkes process with $\mu = 10$, on the right, the event distribution.

In most cases, the motivation of study of point processes is counting the events, but in our case we also are interested in the time of occurrence of the events which will let us define bursts or avalanches of activity that we will use to describe the dynamics of the system. As we have said previously $\alpha = 1$, $\beta = 1$, making n the parameter that controls the weigth of the self-excitation. Essentially, n and α play the same role, so we will use them indistinctly. The previous figures showed the dynamics for $n = 1$, making the system critical as a critical branching process PONER ALGO AQUÍ EN REFERENCIA A LA SECCIÓN ANTERIOR [5]. , but we will also study the dynamics for $n = 2$ and two coupled Hawkes processes.

Hawkes processes with $n = 2$

In this case, the self-excitation does not lead to a critical dynamic, but to a supercritical dynamic. In this situation, the time until the first event occur is given in mean by μ^{-1}, after it occurs, an avalanche of activity starts as shown in Figure 1.7.

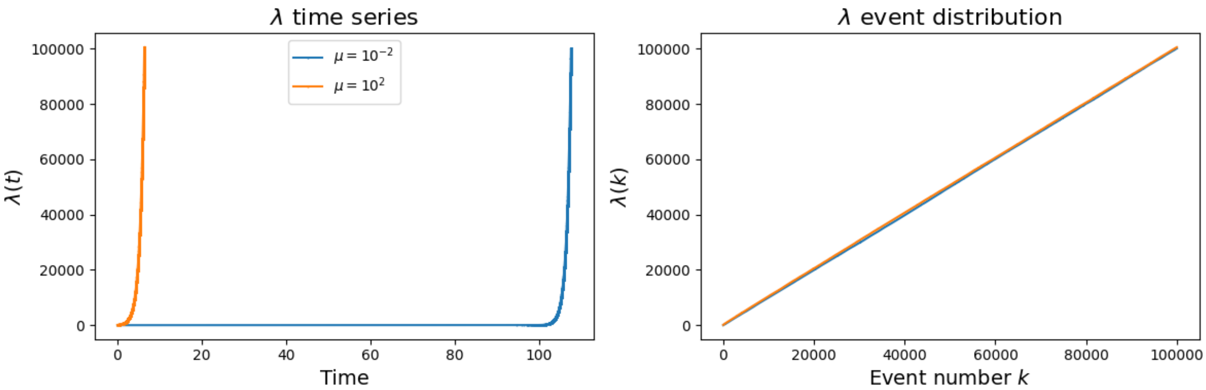
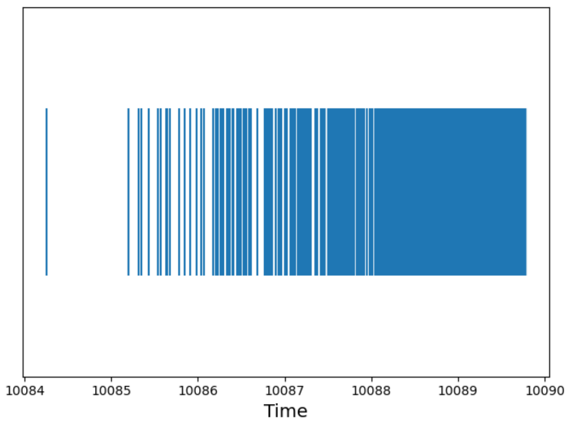
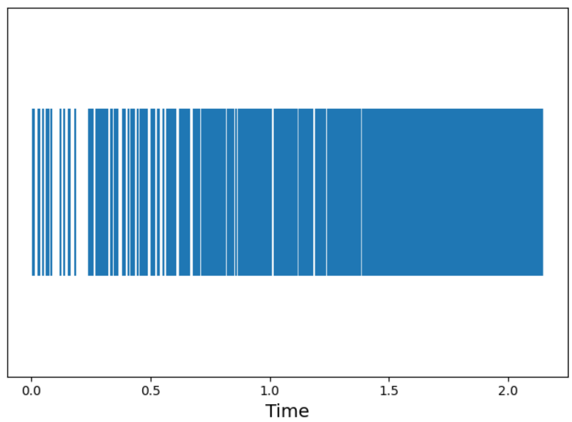


Figure 1.7. On the left, a temporal series of $K = 10^5$ events of a Hawkes process with $\mu \in \{10^{-2}, 10^2\}$ and $n = 2$, on the right, the λ distribution versus the event number.

As expected, after the first event, we have an exponential increase in time and a linear increase in the number of events for the rate. This behaviour will change also the structure of the raster plots, which are where we will get the dynamics information. In Figures 1.8(a) and 1.8(b) two raster plots are shown for the two different background intensities.



(a) Background rate $\mu = 10^{-4}$.



(b) Background rate $\mu = 10^2$.

Figure 1.8. Raster plots of Hawkes processes with $n = 2$. In both cases, the number of events is $K = 10^3$, they also have the same structure, the first event, on average, happens in $\mu^{-1} a.u.$ of time; then, an avalanche of events takes place.

Coupled Hawkes processes

With the previous knowledge, we can model an isolated neuron working in different regimes, but as we know, the brain is composed of an enormous amount of neurons connected to each other forming a network. Moreover, neurons can be classified into two kinds, excitatory and inhibitory neurons. To get closer to modelling the brain, we will also generate two coupled Hawkes processes, one corresponding to an excitatory population and the other to an inhibitory population or just an excitatory and inhibitory neuron. Both populations (neuron) will have a background rate μ_E and μ_I, and they will be able to interact with each other and with themselves, the "strength" of the interactions will be controlled by the parameters n_{EE}, n_{EI}, n_{IE} and n_{II}. In most cases, the auto-inhibition can be considered negligible. These interactions are illustrated in Figure 1.9.

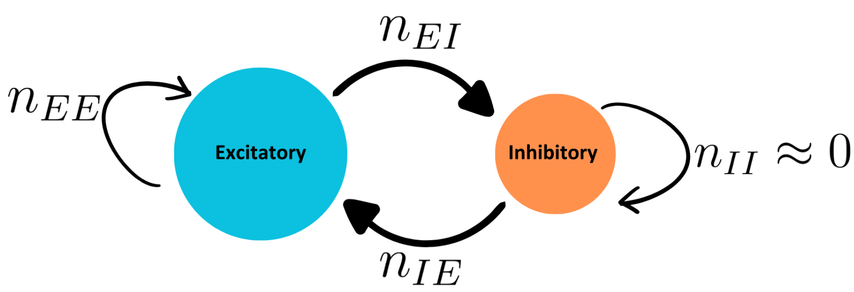


Figure 1.9. Scheme of the interaction between the excitatory and inhibitory populations. As illustrated, the excitatory population can excite itself and the inhibitory population, while the inhibitory population can only inhibit the excitatory population, because on most cases the auto-inhibition is negligible [9].

In a similar way to the previous cases, the background rates will only change time until the first event occurs, but signals will only rely on the "strength" $N_{i,j}$ of the interactions. Studying this parameter space, the different phases and the respective avalanche statistics could be a Master's thesis in itself. That is why we will only look at a few parameter configurations.

Chapter 2

Objectives

Ordenar los objetivos una vez escrito el trabajo para que coincidan con como se presenta.

The main objectives of this Master's thesis are:

- To understand what Hawkes processes are, where we can find them, how to generate them computationally and relate them with neuroscience.
- To understand the importance of time binning and reproduce the results of the original paper [5] and compare them with the results obtained in this work.
- ¿Criticality?
- To study the behaviour of a self-exciting process with $n = 2$ and compare it with the case $n = 1$.
- To study the behaviour of an inhibitory and excitatory neuron coupled.

Chapter 3

Methodology

In the following sections, the methodology for data generation, managment and analysis will be presented. To adress these issues, we will use Python [10, 11] due to its versatility and the wide range of libraries available. The two used will be NumPy [12] and Matplotlib [13] for the visualization.

3.1 Time series factory

The first step is the generation of time series, there are two ways to do this: the slow one and the fast one. The first one is discretizing the time and calculating the rate at each time step according with Eq 1.2, then accept or reject the event if $p < \lambda \cdot dt$ for a random number $p \in \mathcal{U}[0,1]$. This method works for small time series, but for large ones is not efficient because the summation of the kernel function has to be done at each time step. The pseudo-code for this method is presented in Algorithm 1.

Algorithm 1 Slow method to generate Hawkes processes.

Require: $t_m ax$, $n_{intervals}$, $\lambda(t_0) = \mu$, p
  $dt \leftarrow \frac{t_{max}}{n_{intervals}}$
  for $i = 0$ to $n_{intervals}$ **do**1
    $\lambda(t_k) \leftarrow \mu + n \sum_{t_i < t_k} \phi(t_k - t_i)$ ▷ $t_i = i \cdot dt$
    if $\lambda(t_k) \cdot dt > p$ **then**
      $t_{event} \leftarrow t_k$
    end if
  end for

The fast method takes advantage of Monte Carlo methods [14] to generate the time series. The idea of this procedure consists in computing the inter-event time instead of the time of the event. To get to the algorithm, we start from the following expression:

$$PDF(\text{inter-event time} = \Delta t) = \lambda(t + \Delta t) e^{-\int_t^{t+\Delta t} \lambda(t')dt'} \tag{3.1}$$

To demonstrate this, we have to take a look at the Figure 3.1 and recall that λ is a probability per unit of time.

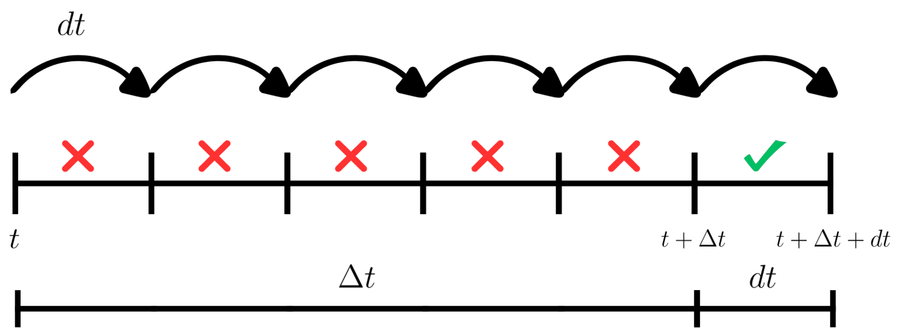


Figure 3.1. Diagram to calculate the cumulative probability of the inter-event time.

The probability per unit of time of having an event in the interval $[t+\Delta t, t+\Delta t+dt]$ is the probability of no events in the interval $[t, t+\Delta t]$ times the probability of happening in the interval $[t+\Delta t, t+\Delta t+dt]$. Putting words into mathematics, we have that the probability of not having an event in the interval $[t, t+\Delta t]$ is:

$$\begin{aligned} P(\text{event} \in [t+\Delta t+dt]) &= (1-\lambda(0)\cdot dt)\,(1-\lambda(dt)\cdot dt)\,(1-\lambda(2dt)\cdot dt)\ldots \\ &= \prod_{k=0} \underbrace{(1-\lambda(kdt)\cdot dt)}_{e^{\ln(1-\lambda(kdt)dt)}} = e^{\sum_{k=0}\ln(1-\lambda(kdt)dt)} = \ldots \quad \text{Using} \ln(1-\varepsilon) \approx -\varepsilon \\ &= e^{-\sum_{k=0}\lambda(kdt)dt} \underbrace{=}_{dt\to 0} e^{-\int_t^{t+\Delta t}\lambda(t')dt'}. \end{aligned} \tag{3.2}$$

Knowing that the probability of having an event in the interval $[t+\Delta t, t+\Delta t+dt]$ is $\lambda(t+\Delta t)dt$, we have:

$$P(\text{event} \in [t+\Delta t, t+\Delta t+dt])\cancel{dt} = \lambda(t+\Delta t)dt \cdot e^{-\int_t^{t+\Delta t}\lambda(t')dt'} \times PDF(\text{inter-event time} = \Delta t)\cancel{dt}. \tag{3.3}$$

Having that we can calculate the inter-event time following the next steps.

$$\text{PDF}\,(\text{inter-event time } = \Delta, t) = \lambda(t+\Delta t) \times \underbrace{e^{\int_t^{t+\Delta t}\lambda(t'dt')}}_{\text{No events during } (t, t+\Delta t)}$$

In order to generate Δt, we will use the inverse transform method [15], therefore we have to calculate the cumulative probability of the inter-event time:

$$\begin{aligned} \text{accum}(\Delta t) &= \int_0^{\Delta t} \text{PDF}\left(\Delta t'\right) d\Delta t' = u \in \mathcal{U}[0,1] \\ &\int_0^{\Delta t} \underbrace{\lambda(t+\Delta t')e^{\int_t^{t+\Delta t'}\lambda(t')dt'}}_{-\frac{d}{d\Delta t'}\left[e^{-\int_t^{t+\Delta t}\lambda t' dt'}\right]} d\Delta t' = u \qquad \text{Using Barrow rule} \\ &- e^{-\int_t^{t+\Delta t}\lambda(t')dt'}\Big|_0^{\Delta t} = 1 - e^{-\int_t^{t+\Delta t}\lambda(t')dt'} = u \qquad \text{Taking logarithms} \\ &\int_t^{t+\Delta t} \lambda(t')dt' = -\ln(1-u) = \ln(\bar{u}) \end{aligned} \tag{3.4}$$

To compute the inter-event time, we have to generate $\bar{u} \sim$ and solve the equation. Having in mind this relation and using Eq 1.4 we have: EN EL LA ECUACIÓN REFERENCIADA Y LA SEGUNDA EXPONENCIAL SE PUEDE PONER $\lambda(t_k^-)$ en lugar de $\lambda(t_k)$? .

$$\begin{aligned} u =& 1 - e^{-\mu(t-t_k)} e^{-(\lambda(t_k)+\alpha-\mu)\cdot \overbrace{\int_{t_k}^{t} e^{-\beta(t')-t_k} dt'}^{-\frac{1}{\beta}\left[e^{-\beta}(t-t_k)-1\right]}} \\ u =& 1 - \underbrace{e^{-\mu(t-t_k)}}_{P(t_{k+1}^{(1)}>t)} \underbrace{e^{-\left[(\lambda(t_k)+\alpha-\mu)\beta^{-1}\left(1-e^{-\beta(t-t_k)}\right)\right]}}_{P(t_{k+1}^{(2)}>t)} \end{aligned} \tag{3.5}$$

Then we apply the composition method [7]. If we take $t_{k+1} = \min(t_{k+1}^{(}1), t_{k+1}^{(}2))$; then $t_{k+1} \sim P(t_{k+1} > t)$, hence:

$$\begin{aligned} \text{Prob}(t_{k+1} = \min\left(t_{k+1}^{(1)}, t_{k+1}^{(2)}\right) \leq t) =& 1 - \text{Prob}\left(\min\left(t_{k+1}^{(1)}, t_{k+1}^{(2)}\right) > t\right) \\ =& 1 - \text{Prob}\left(t_{k+1}^{(1)} > t\right) \cdot \text{Prob}\left(t_{k+1}^{(2)} > t\right) \end{aligned} \tag{3.6}$$

where we have used that the probability that the smaller is greater than t is that each separately is greater than t because both have to be greater than t. As we can see the expressions in Eqs 3.5 and 3.6 are the same, so we can use the composition method to generate the inter-event time. Then, the algorithm to generate the inter-event time is:

1. Generate $t_{k+1}^{(1)} \sim P\left(t_{k+1}^{(1)} > t\right) = e^{-\mu(t-t_k)}$ using

$$P\left(t_{k+1}^{(1)} \leq t\right) = \underbrace{1 - \underbrace{e^{-\mu(t-t_k)}}_{\bar{u_1} \in \mathcal{U}[0,1]}}_{1-\bar{u_1}=u_1} = u_1 \in \mathcal{U}[0,1]$$

   This is done by generating $u_1 \in \mathcal{U}[0,1]$ and solving the equation.

$$\begin{aligned} u_1 &= 1 - e^{-\mu\left(t_{k+1}^{(1)}-t_k\right)} \\ \ln(u_1) &= -\mu\left(t_{k+1}^{(1)} - t_k\right) \Rightarrow t_{k+1}^{(1)} = t_k - \frac{\ln(u_1)}{\mu} \end{aligned} \tag{3.7}$$

2. Generate $t_{k+1}^{(2)} \sim P\left(t_{k+1}^{(2)} > t\right) = e^{-\left((\lambda(t_k)+\alpha-\mu)\beta^{-1}\left(1-e^{-\beta\left(t_{k+1}^{(2)}-t_k\right)}\right)\right)}$ in a similar way as before:

$$\begin{aligned} u_2 &= 1 - e^{-\left((\lambda(t_k)+\alpha-\mu)\beta^{-1}\left(1-e^{-\beta\left(t_{k+1}^{(2)}-t_k\right)}\right)\right)} \\ -\ln(u_2) &= \left((\lambda(t_k) + \alpha - \mu)\,\beta^{-1}\left(1 - e^{-\beta\left(t_{k+1}^{(2)}-t_k\right)}\right)\right) \\ 1 + \frac{\beta \ln u_2}{\lambda(t_k) + \alpha - \mu} &= e^{-\beta\left(t_{k+1}^{(2)}-t_k\right)} \\ t_{k+1}^{(2)} &= t_k - \beta^{-1} \ln \underbrace{\left(1 + \frac{\beta \ln u_2}{\lambda(t_k) + \alpha - \mu}\right)}_{\text{This number must be positive}} \end{aligned} \tag{3.8}$$

3. Choose $t_{k+1} = \min\left(t_{k+1}^{(1)}, t_{k+1}^{(2)}\right)$

4. Calculate the rate at t_{k+1} using Eq 1.3 and go back to step 1.

With this method, we can generate time series efficiently. The pseudo-code for this method is presented in Algorithm 2.

Algorithm 2 Algorithm to generate K Hawkes events.

Require: α, β, $\lambda(t_0) = \mu$, K
  for $k = 0$ to K **do**
    $u_1, u_2 \leftarrow \mathcal{U}[0,1]$
    $t_{k+1}^{(1)} \leftarrow \dfrac{\ln(u_1)}{\mu}$
    $t_{k+1}^{(2)} \leftarrow \beta^{-1} \ln\left(1 + \dfrac{\beta \ln u_2}{\lambda(t_k) + \alpha - \mu}\right)$
    $t_{k+1} \leftarrow \min\left(t_{k+1}^{(1)}, t_{k+1}^{(2)}\right)$
    $\lambda(t_{k+1}) \leftarrow \mu + e^{-\beta(t_{k+1}-t_k)}\left(\lambda(t_k) - \mu + n\right)$
  end for

To conclude generation of time series section, we can generalise the algorithm in order to generate M Hawkes processes coupled [7, 8]. The essence of the algorithm is the same as the one presented. First, we generate the inter-event time for the excitatory population and the inhibitory population, after that, we choose the minimum of both and update the rates of both populations according to the event that has just occurred. Mathematically it is expressed as follows:

1. Generate $\Delta_{k+1} = \min\left\{\Delta_{k+1}^{(1)}, \Delta_{k+1}^{(2)}\right\}$ with $\Delta_{k+1}^{(j)} = t_{k+1}^{(j)} - t_k^{(j)}$ generated as in Eqs 3.7 and 3.8.

$$\Delta_{k+1}^{(j)} = \min\left\{ -\frac{\ln\left(u_1^{(j)}\right)}{\mu_j}, -\beta_j^{-1} \ln\left(\underbrace{1 + \frac{\beta_j \ln u_2^{(j)}}{\lambda_j\left(t_k^{(j)}\right) + \alpha_j - \mu_j}}_{g_j}\right)\right\} \tag{3.9}$$

Note that g_j must be positive, otherwise, take the other term.

2. Once we have the process (l), we update the time for the following event as $t_{k+1} = t_k + \Delta_{k+1}^{(l)}$.

3. Update the rates for the excitatory and inhibitory populations as follows:

$$\lambda_j(t_{k+1}) = \mu_j + e^{-\beta_j(t_{k+1}-t_k)}\left(\lambda_j(t_k) - \mu_j + \alpha_{l\to j}\right) \qquad \text{with } j = 1, 2 \tag{3.10}$$

The pseudo-code for this method is presented in Algorithm 3.

Algorithm 3 Algorithm to generate K Hawkes events for two coupled processes.

Require: α_{11}, α_{12}, β_1, μ_1, α_{22}, α_{21}, β_2, μ_2, K

for $k = 0$ to K **do**

$u_1^{(1)}, u_2^{(1)}, u_1^{(2)}, u_2^{(2)} \leftarrow \mathcal{U}[0, 1]$

$$\Delta_{k+1}^{(1)} \leftarrow \min\left(-\frac{\ln\left(u_1^{(1)}\right)}{\mu_1}, -\beta_1^{-1}\ln\left(1+\frac{\beta_1 \ln u_2^{(1)}}{\lambda_1(t_k)+\alpha_{11}-\mu_1}\right)\right)$$

$$\Delta_{k+1}^{(2)} \leftarrow \min\left(-\frac{\ln\left(u_1^{(2)}\right)}{\mu_2}, -\beta_2^{-1}\ln\left(1+\frac{\beta_2 \ln u_2^{(2)}}{\lambda_2(t_k)+\alpha_{22}-\mu_2}\right)\right)$$

$l \leftarrow \arg\min\left(\Delta_{k+1}^{(1)}, \Delta_{k+1}^{(2)}\right)$

$t_{k+1} \leftarrow t_k + \Delta_{k+1}^{(l)}$

$\lambda_1(t_{k+1}) \leftarrow \mu_1 + e^{-\beta_1(t_{k+1}-t_k)}\left(\lambda_1(t_k) - \mu_1 + \alpha_{l\to 1}\right)$

$\lambda_2(t_{k+1}) \leftarrow \mu_2 + e^{-\beta_2(t_{k+1}-t_k)}\left(\lambda_2(t_k) - \mu_2 + \alpha_{l\to 2}\right)$

end for

3.2 When physics and cooking merge

Now we have a method to generate the main ingredient, time series. In order to cook (analyze) them, our tools will be Python libraries and a control parameter, which in our case will be a resolution parameter $\Delta > 0$ that will allow us to identify clusters of activity. Assuming that we have a time series with K events that happen in times $\{t_1, \ldots, t_K\}$. Each event starts a cluster and the following event will be part of this cluster if the time between both events is less than Δ and so on for the rest. We define the size of the cluster as the number of events in the interval $[t_{first}, t_{last}]$ and its duration as $t_{last} - t_{first}$. The extreme cases are when Δ is smaller than the minimun inter-event time, where each event is a cluster of size 1 and duration 0. The other extreme is when Δ is greater than the largest inter-event time, where all the events are in the same cluster of size K and duration $t_K - t_1$. Between these two extremes, we will have different regimes of the process. Our recipe will be the phase diagram, specifically the percolation diagram, where we will plot the percolation strength P_∞ as a function of the resolution parameter Δ. The percolation strength is defined as the fraction of events that are in the largest cluster over the total number of events. Three different set of parameters will be used to generate the time series in order to compare them. The parameters α and β will be fixed to unless otherwise stated, the other parameters are shown in Table 3.1. Once we got our recipe (the percolation diagram), we should be able to identify the critical points and the different regimes of the process. By carefully choosing Δ from the percolation diagram, we will be able to identify the different regimes with their respective power law exponentes.

Table 3.1. Configuration of the parameters for the simulations of the article [5].

Configuration	μ	n
First	1	0
Second	10^{-4}	1
Third	10^{2}	1

The percolation diagram will be generated by generating 1000 time series for each configuration and calculating the percolation strength for each one because in general they are not stationary processes as we can observe in Figure 3.2.

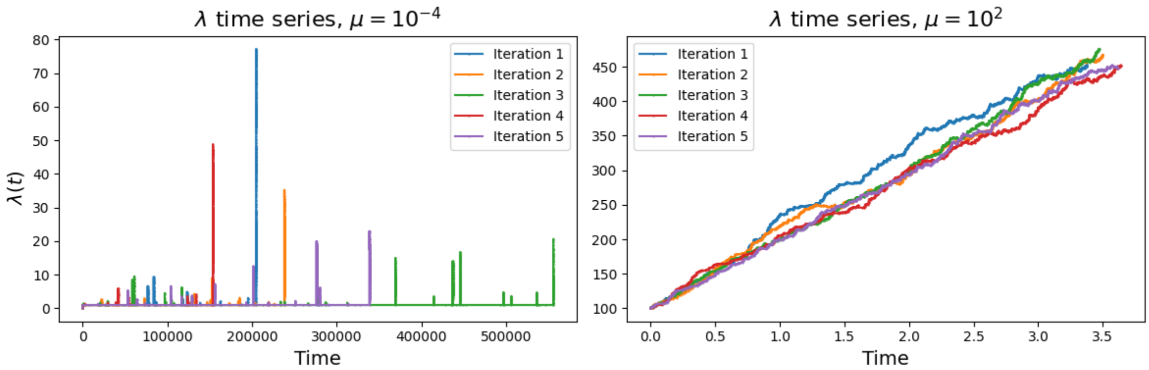


Figure 3.2. Five a temporal series of $K = 10^5$ events of Hawkes processes with $\mu = 10^{-4}$ on the left side and $\mu = 10^2$ on the right one.

Beginning with the first configuration, we have an homogeneous Poisson process which we know that its interevent time will be distributed randomly with a probability of having an inter-event time x_i given by $P(x_i) = \mu e^{-\mu x_i}$. Consequently, two consecutive events will be a part of a cluster fixing the resolution parameter to Δ with a probability of

$$P(x_i \leq \Delta) = 1 - e^{-\mu\Delta} \qquad \forall i. \tag{3.11}$$

This represents the probability in a homogeneous 1D percolation model [16], where we can identify a non percolant phase and a percolant phase separated by the critical point Δ^*. We can calculate this parameter if we know the maximum inter-event time of the time series. Let us assume that our time series has K events, therefore, it will percolate if the condition we have just stablished is satisfied. We can calculate this threshold as the average of the maximum inter-event time in K samples from the inter-event time distribution solving the following equation:

$$\begin{aligned}
&K \int_{\Delta^*}^{\infty} P(x)dx = 1 \\
&K \int_{\Delta^*}^{\infty} \mu e^{-\mu x} dx = 1 \\
&-K \left[e^{-\mu x}\right]_{\Delta^*}^{\infty} = K \left[e^{-\mu\Delta^*} - \cancelto{0}{e^{-\mu\infty}}\right] = 1 \\
&K e^{-\mu\Delta^*} = 1 \\
&\Delta^*(K) = -\frac{\ln(K)}{\mu}
\end{aligned} \tag{3.12}$$

For the other two configurations, on both we have a self-exciting process with $n = 0$, which means that we have a critical dynamical regime as we shall see later but with different background rates, one much smaller than 1 and the other much greater than 1. This fact will be reflected in the percolation diagram. We will not approach this cases from a theoretical point of view but from a graphic one. With the second configuration, as we have seen in Figure 1.5 if the condition $\mu \ll 1$ is satisfied, we will have a bursty structure in the time series. Due to the low background rate, the events are less likely to occur, but when they do, they tend to form avalanches of activity thanks to the self-excitation. This will be reflected in the percolation diagram as a first phase transition at a critical point Δ_1^* when Δ is of the order of the average cluster size. Then, a second phase transition will occur at a critical point Δ_2^* when Δ is greater than the greatest inter-event time. This phenomena is illustrated in Figure 3.3.

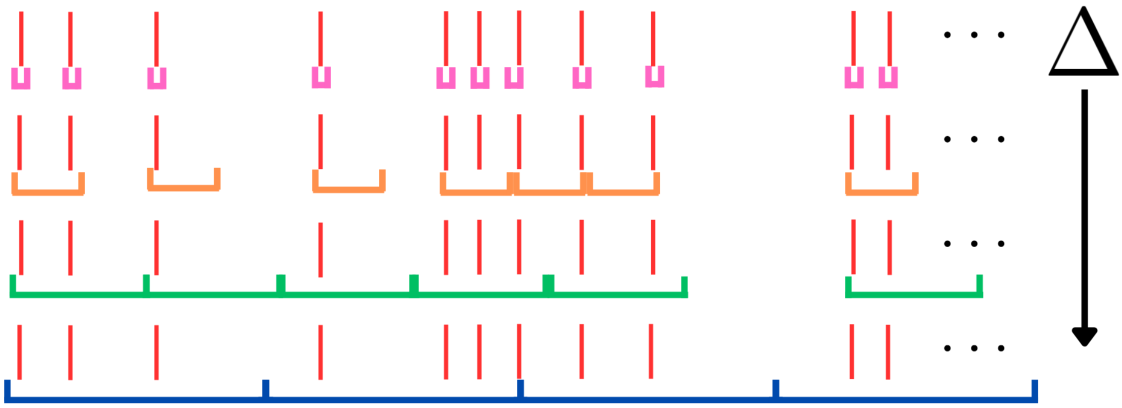


Figure 3.3. Diagram for $\mu \ll 1$. Red lines represent the events, clusters are coloured. As we can see, we have two regimes, one when Δ is of the order of the average cluster size and another when is of the order of the inter-event time where the system percolates.

On the other hand, when $\mu \gg 1$ events occur more frequently, without making the bursty structure of Figure 1.5, but making a more regular structure as illustrated in Figure 1.6. This will be reflected in the phase diagram as a single phase transition at a critical point Δ^* when Δ is of the order of the average cluster size. This phenomena is illustrated in Figure 3.4. Note the absence of a time scale in both diagram, they are diagrams for the explanation of the phase diagram.

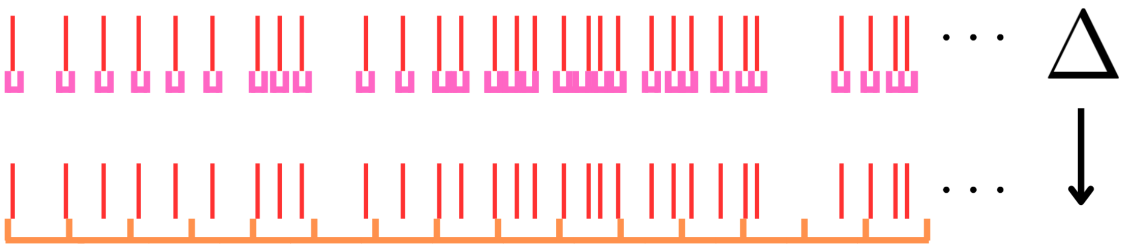


Figure 3.4. Diagram for $\mu \gg 1$. Red lines represent the events, clusters are coloured. In this situation, events occur more regularly, resulting in a unique transition corresponding the case of Δ is similar to the inter-event time producing the system percolation.

Chapter 4

Results

This section provides the main results of the investigation. The main functions used to obtain these results are shown in the Appendix A. The results are divided into three sections:

1. First, the results reproduced from the original paper [5] are presented.
2. Secondly, we will present the same analysis for the case of a Hawkes process with $n = 2$.
3. Finally, we will study the behaviour of two Hawkes processes couples, one representing an excitatory neuron and the other an inhibitory neuron.

4.1 Results from the original paper

The structure for the three sections will be the same. First, we will obtain the phase diagram for the percolation strength P_∞ versus our control parameter, the resolution parameter Δ, obtaining the critical(s) point(s) Δ_i^*. Having these in knowledge, the avalanche statistics for the size and duration will be studied for different regions of the phase diagram.

As previously stated, the first result is the percolation phase diagram, shown in Figure 4.1. It displays the percolation strength P_∞, which is the number of events in the largest cluster divided by the total number of events of the time series versus the resolution parameter Δ. We will generate several time series, compute the percolation strength for each one and take the average value.

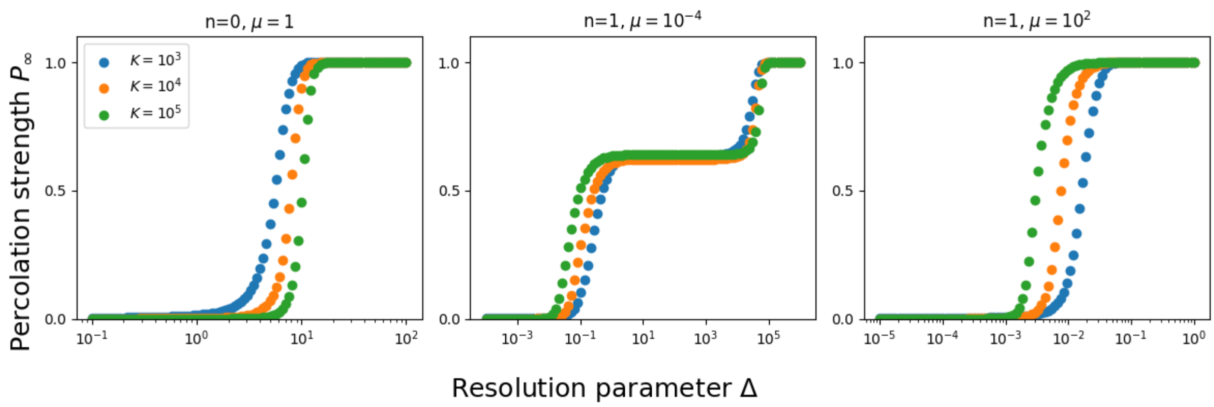


Figure 4.1. Percolation phase diagrams for different event number K taking average values of $R = 1000$ realizations.

The first plot configuration is a homogeneous Poisson process with rate $\mu = 1$ which we have overviewed in Section 1.2.1 and has a pseudocritical threshold at $\Delta^*(K) = \frac{\ln(K)}{\mu}$ as we have demonstrated in Section 3.2. Due to the fact of finite size of the time series, the transition is discontinuous at the threshold, as expected for 1D percolation [16].

Now, we contemplate Hawkes processes, for the first case $\left(\mu = 10^{-4}\right)$, we can observe a double discontinuous transition. The first one at Δ_1^* and the second one at Δ_2^*. As we are going to see with the avalanche statistics, the first transition is associated with the universality class of 1D percolation whose exponents are $\alpha = \tau = 2$. On the other hand, the second transition is associated with the universality class of mean-field branching process whose exponents are $\alpha = 3/2$ and $\tau = 2$. This double transition is also compatible with the fact mentioned in Figure 3.3. We can also observe that the plateau between the two transitions is wider as the K increases as expected. For the second case $\left(\mu = 10^2\right)$, similarly to the first one, we have a single discontinuous transition at Δ_1^* associated with the universality class of 1D percolation as well, this phenomenon is also compatible with the one shown in Figure 3.4.

Another interesting analysis to characterize the phases is studying the susceptibility χ. In this case, it is defined by Eq 4.1.

$$\begin{aligned}\chi =& \frac{\langle S_M^2 \rangle - \langle S_M \rangle^2}{\langle S_M \rangle} \\ =& K \cdot \frac{\langle P_\infty^2 \rangle - \langle P_\infty \rangle^2}{\langle P_\infty \rangle} \\ =& K \cdot \frac{\sigma^2\left(P_\infty\right)}{\langle P_\infty \rangle}\end{aligned} \tag{4.1}$$

The susceptibility (normalized to the number of events) is shown in Figure 4.2. For the Poisson process, we see that the susceptibility has a peak at the the threshold $\Delta^*(K)$, then it vanishes as expected. For the Hawkes case with $\mu = 10^{-4}$, we observe that χ has a peak at the critical point Δ_1^*, then we have a critical behaviour where the susceptibility is not zero at the plateau $[\Delta_1^*, \Delta_2^*]$ and finally, it vanishes at the second critical point Δ_2^*. Finally, for the Hawkes case with $\mu = 10^2$ and likewise the Poisson process, the susceptibility has a divergence at the critical point Δ_1^* and then it vanishes.

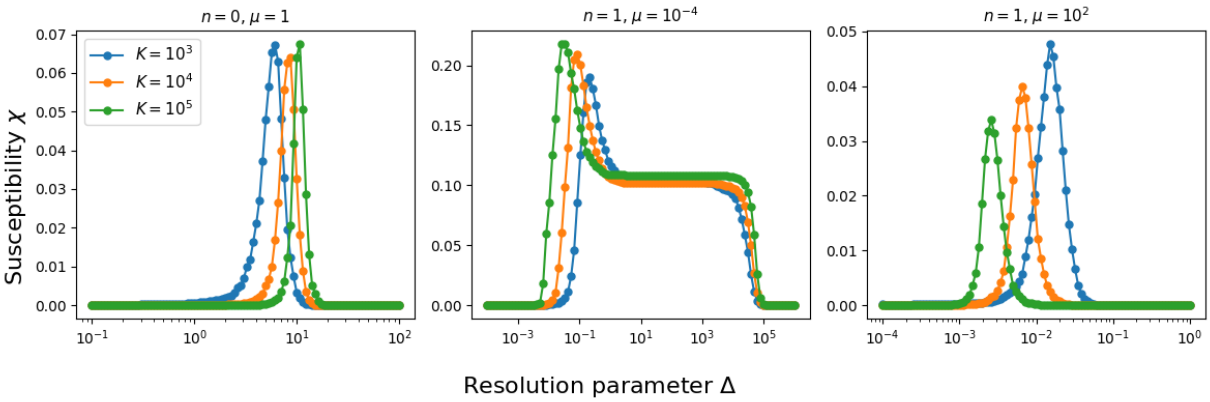


Figure 4.2. Susceptibility χ normalized to the number of evetns K, for different event number K and taking average values for $R = 1000$ realizations.

Once we have the phase diagram, we can study avalanche statistics, but first, we need to obtain the thresholds Δ_1^* and Δ_2^* from the phase diagram. The article [5] provides the following formulas to compute these thresholds for the Hawkes process with $\mu = 10^{-4}$ and:

$$\Delta_1^* \simeq \frac{\ln(K)}{\langle\lambda\rangle} = \frac{\ln(K)}{\mu + \sqrt{2\mu K}} \tag{4.2}$$

$$\Delta_2^* = \frac{\ln(K)}{\mu} \tag{4.3}$$

and for $\mu = 10^2$:

$$\Delta_1^* = \frac{\ln(K)}{\mu} \tag{4.4}$$

Bearing this in mind and the definitions of the size and duration of avalanches established in the previous chapter, we can study the avalanches for the different regions of the phase diagram. We just are going to show the results for $\mu = 10^{-4}$ and for $\mu = 10^2$ in Figure 4.3. The Poisson process behaviour can be found in [16, 17].

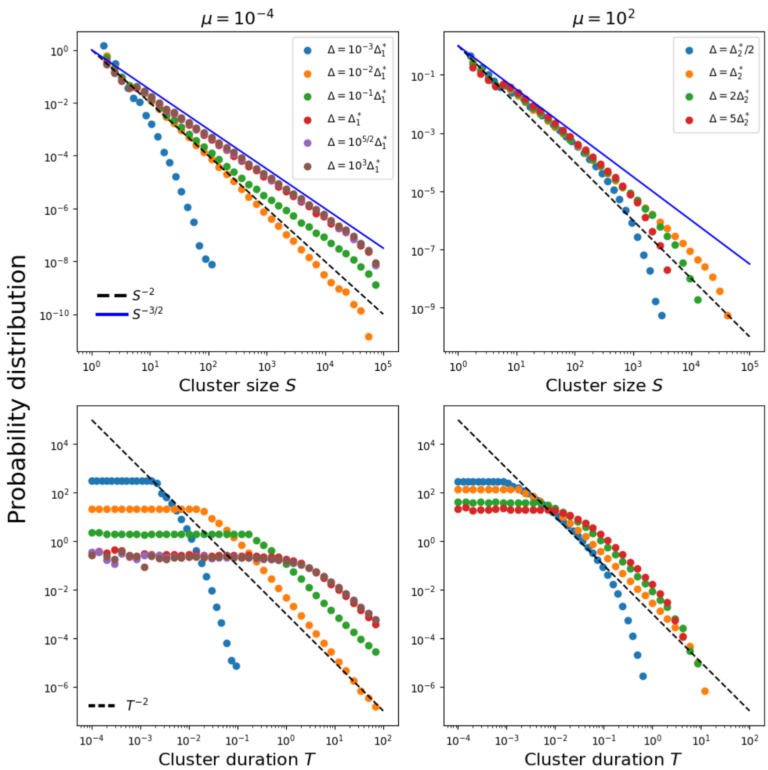


Figure 4.3. Avalanche analysis for Hawkes process with $n = 1$, $K = 10^5$ events averaged over $R = 1000$ realizations.

As a consequence of the huge simulation time of time series of $K = 10^8$ events, we have only studied the avalanches for $K = 10^5$, moreover, we have taken other criterion to obtain the histograms. Instead of considering $C = 10^7$ clusters, we have obtained the histograms of $R = 1000$ time series. This leads to a different amount of clusters for each value of Δ, nevertheless, we have obtained equivalent and reliable results.

For $\mu = 10^{-4}$, the probability distribution of the cluster size and duration shows three different behaviours. For $\Delta \ll \Delta_1^*$, the behaviour is subcritical, leading to to a exponential decay for the size and duration. While we increase Δ, we reach the critical point where the exponents are $\alpha = \tau = 2$ compatible with the universality class of 1D percolation. After that, we reach the plateau $[\Delta_1^*, \Delta_2^*]$ where we have a crossover to the universality class of mean-field branching process and 1D percolation. Finally, for $\Delta \to \Delta_2^*$, we obtain the universality class of mean-field branching process exponents $\alpha = 3/2$ and $\tau = 2$. For $\mu = 10^2$, the plots show a power-law distribution for both cluster size and duration with exponents $\alpha = \tau = 2$ corresponding to the universality class of 1D percolation as we have mentioned before.

Note that we have reproduced the same behaviour, but for other values of Δ, specifically, for two order of magnitude less than article value. This is due to the fact that Eq. 4.2 needs the assumption of large time series, condition which is not fulfilled in our case. We can illustrate this difference for example in the susceptibility diagram, where the peak should be at Δ_1^*, but in our case, it is at $\Delta_1^*/100$ as shown in Figure 4.4.

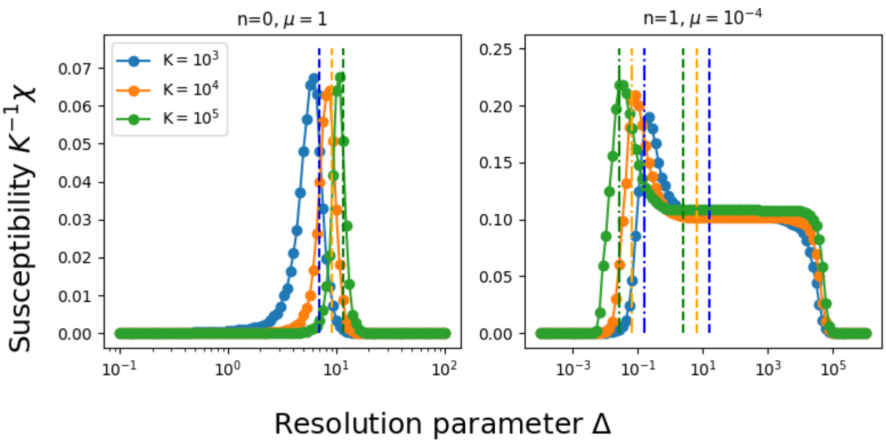


Figure 4.4. At the left, the vertical dashed lines represent the critical points $\Delta^*(K)$ for the Poisson process. At the right, the vertical dashed lines represent the critical points $\Delta_1^*(K)$ given by Eq. 4.2 and the dotted dashed lines the $\Delta_1^*/100$.

4.2 Results for n=2

In the article, the authors have studied a process which is critical itself because the parameter n is fixed to $n = 1$. We have studied the case $n = 2$ to see if the process is still critical. In the Figure 4.5 two event series for $n = 1$ and $n = 2$ are shown.

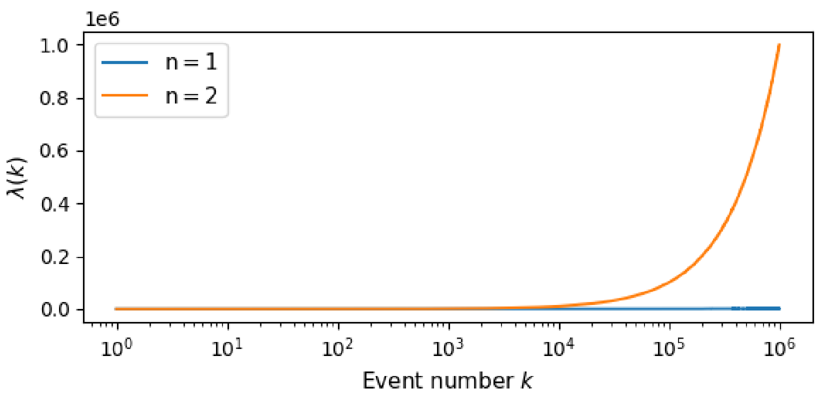


Figure 4.5. Event series for $n = 1$ and $n = 2$.

As presented in the figure above, the rate of the process for $n = 2$ explodes in comparison with the rate for $n = 1$. This is due to the fact that choosing $n = 2$ makes the process supercritical. Similarly to the previous section, the first step is obtaining the phase diagram in order to distinguish the regimes.

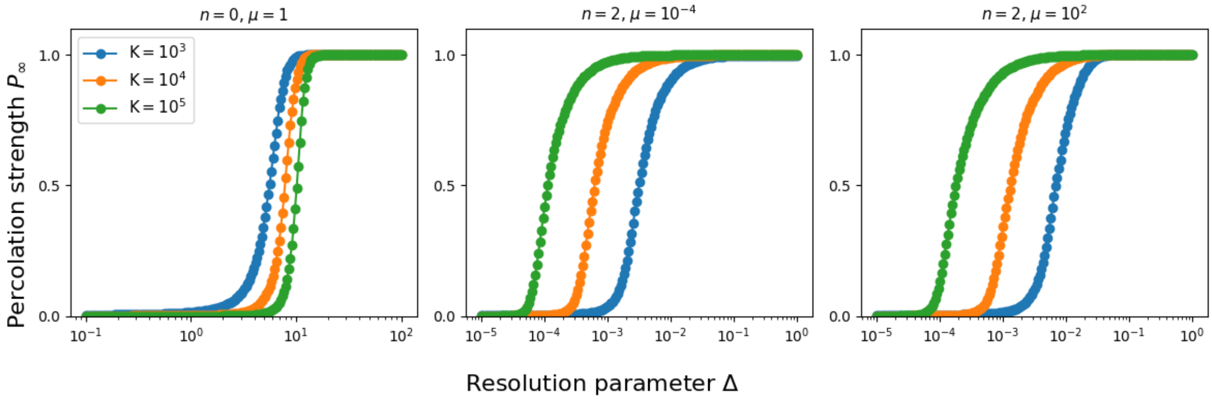


Figure 4.6. Percolation phase diagrams for a Hawkes process with $n = 2$.

In this case, Eqs. 4.2,4.3 are not valid because they were derived for $n = 1$. Therefore, we will obtain this parameter graphically from the phase diagrams shown in Figure 4.6. We will stablish Δ^* at the resolution parameter where the percolation strength $P_\infty = 0.5$, consequently, $\Delta^* \approx 10^{-4}$ for both cases. Alike the previous section, in Figure 4.7 the susceptibility is shown.

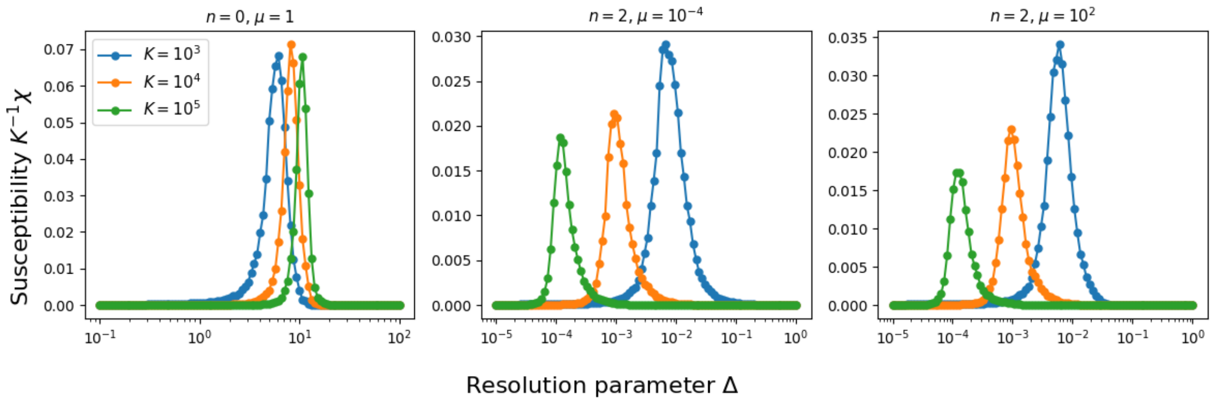


Figure 4.7. Susceptibility χ normalized to the number of evetns K, for different event number K and taking average values for $R = 1000$ realizations.

As we can recognize from both figures, now we have a single transition for $\mu = 10^{-4}$ and $\mu = 10^2$, in principle corresponds to 1D percolation, ergo, the exponents for the size and duration should be $\alpha = \tau = 2$. Identically to the case of $n = 1$, we have studied the avalanches for $K = 10^5$ events and $R = 1000$ realizations to obtain the histograms. The statistics of the avalanches are shown in Figure **??**.

4.3 Inhibitory and excitatory neurons coupled

Chapter 5

Conclusions

Bibliography

[1] Gerardo Chowell et al. "Mathematical models to characterize early epidemic growth: A review". In: *Physics of life reviews* 18 (2016), pp. 66–97.

[2] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. "Statistical physics of social dynamics". In: *Reviews of modern physics* 81.2 (2009), p. 591.

[3] Sandro Azaele et al. "Statistical mechanics of ecological systems: Neutral theory and beyond". In: *Reviews of Modern Physics* 88.3 (2016), p. 035003.

[4] Alan J McKane. *Stochastic Processes.* 2009.

[5] Daniele Notarmuzi et al. "Percolation theory of self-exciting temporal processes". In: *Physical Review E* 103.2 (2021), p. L020302.

[6] Kiyoshi Kanazawa and Didier Sornette. "Ubiquitous power law scaling in nonlinear self-excited Hawkes processes". In: *Physical review letters* 127.18 (2021), p. 188301.

[7] Angelos Dassios and Hongbiao Zhao. "Exact simulation of Hawkes process with exponentially decaying intensity". In: (2013).

[8] Patrick J Laub, Young Lee, and Thomas Taimre. *The elements of Hawkes processes.* Springer, 2021.

[9] Felipe Yaroslav Kalle Kossio et al. "Growing critical: self-organized criticality in a developing neural system". In: *Physical review letters* 121.5 (2018), p. 058301.

[10] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* "" O'Reilly Media, Inc.", 2012.

[11] Jake VanderPlas. *Python data science handbook: Essential tools for working with data.* " O'Reilly Media, Inc.", 2016.

[12] Travis Oliphant et al. *NumPy.* `https://numpy.org/`. Accessed: 2024-06-21. 2024.

[13] John D. Hunter et al. *Matplotlib.* `https://matplotlib.org/`. Accessed: 2024-06-21. 2024.

[14] Adrian Barbu, Song-Chun Zhu, et al. *Monte Carlo Methods.* Vol. 35. Springer, 2020.

[15] Raúl Toral and Pere Colet. *Stochastic numerical methods: an introduction for students and scientists.* John Wiley & Sons, 2014.

[16] Dietrich Stauffer and Ammon Aharony. *Introduction to percolation theory.* Taylor & Francis, 2018.

[17] D Stauffer and C Jayaprakash. "Critical exponents for one-dimensional percolation clusters". In: *Physics Letters A* 64.5 (1978), pp. 433–434.

Appendix A

Python scripts

REVISAR LOS CÓDIGOS PARA QUE ESTÉN ACTUALIZADOS

CAMBIAR FUNCIONES.PY PARA QUITAR LA GENERATE_SERIES_PERC

AÑADIR FUNCIÓN DE POISSON Y GITHUB EN LA bibliografía Here are the main functions used in the project. The first one is the algorithm used to simulate the inter-event time of a Hawkes process with exponential kernel. The second one is the function for the generation of time series of these Hawkes processes. The third one is the function associated to the calculation of the percolation strength, P_∞, for the phase diagrams. The fourth one is the function to identify the clusters size and duration distribution. The fifth one is the function to generate the inter-event time of a bivariate Hawkes process with exponential kernel. The last one is the function to generate the time series of the bivariate Hawkes process.

Script A.1. Script with the main functions.

```
import numpy as np

def algorithm(rate, mu, n):
    """
    Algorithm that computes interevent times and Hawkes intensity for a self-exciting
    process

    ## Inputs:
    rate: Previous rate
    mu: Background intensity
    n: Weight of the Hawkes process

    ## Outputs: rate x_k, x_k
    x_k: Inter-event time
    rate_tk: Intensity at time tk
    """
    # 1st step
    u1 = np.random.uniform()
    if mu == 0:
        F1 = np.inf
    else:
        F1 = -np.log(u1) / mu

    # 2nd step
    u2 = np.random.uniform()
    if (rate - mu) == 0:
        G2 = 0
```

```
    else:
        G2 = 1 + np.log(u2) / (rate - mu)


    # 3rd step
    if G2 <= 0:
        F2 = np.inf
    else:
        F2 = -np.log(G2)

    # 4th step
    xk = min(F1, F2)

    # 5th step
    rate_tk = (rate - mu) * np.exp(-xk) + n + mu
    return rate_tk, xk

def generate_series(K, n, mu):
    """
    Generates temporal series for K Hawkes processes

    ##Inputs:
    K: Number of events
    n: Strength of the Hawkes process
    mu: Background intensity

    ##Output:
    times_between_events: time series the inter-event times
    times: time series the events
    rate: time series for the intensity
    """
    times_between_events = [0]
    rate = [mu]
    for _ in range(K):
        rate_tk, xk = algorithm(rate[-1], mu, n)
        rate.append(rate_tk)
        times_between_events.append(xk)
    times = np.cumsum(times_between_events)
    return times_between_events, times, rate

def calculate_percolation_strength(times_between_events, deltas):
    """
    Calculate the percolation strength for a given set of deltas (resolution parameters)

    ## Inputs:
    times_between_events: time series of interevent times
    deltas: list of resolution parameters

    ## Output:
    percolation_strengths: list of percolation strengths
    """

    percolation_strengths = []

    for delta in deltas:
        cluster_sizes = []
        current_cluster_size = 1 # The first event is always a cluster

        for time in times_between_events:
            if time < delta:
                current_cluster_size += 1
            else:
```

```python
                if current_cluster_size > 1: # Only consider clusters with more than one event
                    cluster_sizes.append(current_cluster_size)
                current_cluster_size = 1 # The next event is always a cluster

        if current_cluster_size > 1: # Consider the last cluster if it ends at the last event
            cluster_sizes.append(current_cluster_size)

        if len(cluster_sizes) != 0:  # Check if cluster_sizes is not empty to avoid errors
            max_cluster_size = max(cluster_sizes)
        else:
            max_cluster_size = 0

        percolation_strengths.append(max_cluster_size / len(times_between_events))

    return percolation_strengths

def identify_clusters(times, delta):
    """
    Identifies clusters in a temporal series given a resolution parameter delta

    ## Inputs:
    times: temporal series
    delta: resolution parameter

    ## Output:
    clusters: list of clusters
    """
    clusters = []
    current_cluster = []
    for i in range(len(times) - 1):
        if times[i + 1] - times[i] <= delta:
            if not current_cluster:
                current_cluster.append(times[i])
            current_cluster.append(times[i + 1])
        else:
            if current_cluster:
                clusters.append(current_cluster)
                current_cluster = []
    return clusters

def model(n_max, mu_E, mu_I, tau, n_EE, n_IE, n_EI, n_II, dt):
    """
    Solve the equations of the mena field model for a given number of iterations n_max

    Inputs:
    n_max: number of iterations
    mu_E: Poisson rate of excitatory neurons
    mu_I: Poisson rate of inhibitory neurons
    tau: characteristic time of the system
    n_EE: influence of excitatory neurons on excitatory neurons
    n_IE: influence of excitatory neurons on inhibitory neurons
    n_EI: influence of inhibitory neurons on excitatory neurons
    n_II: influence of inhibitory neurons on inhibitory neurons
    dt: time step

    Outputs:
    time: time series
    t_events_E: times of events of excitatory neurons
    t_events_I: times of events of inhibitory neurons
```

```
    rates_E: rates of excitatory neurons
    rates_I: rates of inhibitory neurons
    """
    n_E = n_I = n = 0
    t_events_E = [0]
    t_events_I = [0]
    rates_E = [mu_E]
    rates_I = [mu_I]
    time = [0]
    while n <= n_max:
        # Excitation neurons
        l_Enew = rates_E[-1]  + dt * (mu_E- rates_E[-1])/tau
        if np.random.uniform() < rates_E[-1]*dt:
            l_Enew += n_EE
            t_events_E.append(time[-1]+dt*np.random.uniform())
            n_E += 1
        if np.random.uniform() < rates_I[-1]*dt:
            l_Enew -= n_IE
            t_events_E.append(time[-1]+dt*np.random.uniform())
            n_E += 1

        # Inhibition neurons
        l_Inew = rates_I[-1] + dt * (mu_I- rates_I[-1])/tau
        if np.random.uniform() < rates_E[-1]*dt:
            l_Inew += n_EI
            t_events_I.append(time[-1]+dt*np.random.uniform())
            n_I += 1
        if np.random.uniform() < rates_I[-1]*dt:
            l_Inew -= n_II
            t_events_I.append(time[-1]+dt*np.random.uniform())
            n_I += 1
        rates_E.append(l_Enew)
        rates_I.append(l_Inew)
        time.append(time[-1]+dt)

        n = n_E + n_I
    return time, t_events_E, t_events_I, rates_E, rates_I

def identify_clusters_model(times, delta):
    """
    Identifies clusters in a temporal series given a resolution parameter delta
    Computes the size and duration of clusters

    ## Inputs:
    times: temporal series
    delta: resolution parameter

    ## Output:
    clusters: list of clusters
    clusters_sizes: list of sizes of clusters
    clusters_times: list of durations of clusters
    """
    clusters = []
    current_cluster = []
    for i in range(len(times) - 1):
        if times[i + 1] - times[i] <= delta:
            if not current_cluster:
                current_cluster.append(times[i])
            current_cluster.append(times[i + 1])
        else:
            if current_cluster:
                clusters.append(current_cluster)
```

```python
                current_cluster = []

    clusters_sizes = [len(cluster) for cluster in clusters]
    clusters_times = [cluster[-1] - cluster[0] for cluster in clusters]
    return clusters, clusters_sizes, clusters_times

def bivariate_algorithm(rate1, rate2, muE, muI, nEE, nII, nEI, nIE):
    """
    Algorithm that computes interevent times and Hawkes intensity for a bivariate Hawkes process

    #Inputs:
    rate1: Previous excitation rate
    rate2: Previous inhibition rate
    nEE: "Strength" of the autoexcitation process
    nII: "Strength" of the autoinhibition process
    nEI: "Strength" of the excitation to the inhibition
    nIE: "Strength" of the inhibition to the excitation
    muE: Background intensity of the excitation
    muI: Background intensity of the inhibition


    #Output: ratex_k, x_k, reaction (0 for excitatory events and 1 for inhibitory events)
    """
    _, xk1 = algorithm(rate1, muE, nEE)
    _, xk2 = algorithm(rate2, muI, nII)

    xks = [xk1, xk2]

    reaction = np.argmin(xks)

    rate1_tk = 0.
    rate2_tk = 0.

    if reaction == 0:
        rate1_tk = (rate1 - muE) * np.exp(-xk1) + nEE + muE
        rate2_tk = (rate2 - muI) * np.exp(-xk1) + nEI + muI
    else:
        rate1_tk = (rate1 - muE) * np.exp(-xk2) + nIE + muE
        rate2_tk = (rate2 - muI) * np.exp(-xk2) + nII + muI


    if rate1_tk <= muE:
        rate1_tk = muE
    if rate2_tk <= muI:
        rate2_tk = muI

    xk = xks[reaction]

    return rate1_tk, rate2_tk, xk, reaction

def generate_series_bivariate(K, nEE, nII, nEI, nIE, muE, muI):
    """
    Generates temporal series for K bivariate Hawkes processes

    ##Inputs:
    K: Number of events
    nEE: "Strength" of the autoexcitation process
    nII: "Strength" of the autoinhibition process
    nEI: "Strength" of the excitation to the inhibition
    nIE: "Strength" of the inhibition to the excitation
    muE: Background intensity of the excitation
```

```
    muI: Background intensity of the inhibition

    ##Output:
    times_between_events: time series the interevent times
    times: time series the events
    rate1: time series for the intensity of process 1 (Excitation)
    rate2: time series for the intensity of process 2 (Inhibition)
    reactions: list the event type (0 for excitation. 1 for inhibition)
    """
    times_between_events = [0]
    rate1 = [muE]
    rate2 = [muI]
    reactions= []
    for _ in range(K):
        rate1_tk, rate2_tk, xk, reaction = bivariate_algorithm(rate1[-1], rate2[-1], muE,
 muI, nEE, nII, nEI, nIE)
        rate1.append(rate1_tk)
        rate2.append(rate2_tk)
        reactions.append(reaction)
        times_between_events.append(xk)
    times = np.cumsum(times_between_events)

    return  times_between_events, times, rate1, rate2, reactions
```