BACHELOR'S DEGREE THESIS

Degree in Mathematics

Title: Phase-responsiveness transmission in a network of quadratic integrate-and-fire neurons

Author: Ferran Arqué Pérez

Advisor: Antoni Guillamon Grabolosa, J. Tomás Lázaro Ochoa

Department: Department of Mathematics

Academic year: 2019-2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

Phase-responsiveness transmission in a network of quadratic integrate-and-fire neurons

Ferran Arqué Pérez

Supervised by Antoni Guillamon Grabolosa and J. Tomás Lázaro Ochoa

January, 2020

Vull agrair als meus tutors, Toni i Tomás, per la seva ajuda a l'hora d'elaborar d'aquest treball i per tot el que he après amb ells. També vull donar les gràcies al David, pels seus consells i per estar sempre disposat a escoltar-me quan tenia dubtes. Finalment, dono les gràcies a la meva família i amics per tot el suport rebut durant els meus estudis.

Abstract

In the study of the dynamics of neuronal networks, it is interesting to see how the interaction between neurons can elicit different behaviours in each individual one. Moreover, this can lead to the population exhibiting collective phenomena that is not intrinsic to a single cell, such as synchronization. In dynamical systems theory, this problem has been tackled through both high-dimensional systems of coupled single-cell models and mean-field models that describe the macroscopic state of the network in terms of the firing rate or the mean membrane potential.

In this project, we work with a large-scale network and a firing-rate model of quadratic integrate-and-fire (QIF) neurons. We study the dynamics of the QIF model and compute its phase response curve (PRC), which is a well-known tool for the analysis of the perturbations of the cell's membrane potential caused by the different stimuli received from the network. Then, we propose an algorithm to describe the population through the PRCs. Our method is able to replicate the same dynamics we observe with the aforementioned models and it also serves us to gain more insight into the transmission of pulses and to explain how a network can maintain a state of synchronized firing. Our results are a positive test that a mean-field model with PRCs could be obtained.

Keywords: mathematical neuroscience, quadratic integrate-and-fire, neuronal network, phase response curve, synchronization.

Resum

En l'estudi de la dinàmica de xarxes neuronals, és interessant veure com la interacció entre neurones pot provocar diferents comportaments en cada una. Això pot portar inclús a que la població mostri una fenomenologia col·lectiva no inherent a cap neurona, com és el cas de la sincronització. En la teoria de sistemes dinàmics, aquest problema s'ha atacat a partir de sistemes d'alta dimensió on s'acoblen models neuronals i també a partir de models de camp mitjà, que descriuen l'estat macroscòpic de la xarxa a partir de la freqüència de descàrrega (*firing-rate*) o del potencial de membrana mitjà.

En aquest projecte, treballem amb una xarxa de gran escala i amb un model de firing-rate de neurones de tipus *quadratic integrate-and-fire* (QIF). Estudiarem la dinàmica del model QIF i calcularem la seva corba de resposta de fase (PRC), que és una eina ben coneguda utilitzada per l'anàlisi de les pertorbacions del potencial de membrana de la cèl·lula, que són causades pels diferents estímuls provinents de la xarxa. Seguidament, proposem un algorisme que descriu la població a partir de les PRCs. El nostre mètode és capaç de replicar la mateixa dinàmica que observem amb els models anteriors i ens serveix per entendre millor la transmissió d'impulsos i per explicar com una xarxa pot mantenir un estat de sincronia. Els resultats obtinguts són un test positiu de què es podria obtenir un model de camp mitjà amb PRCs.

Contents

1 Introduction

A neuron is an excitable cell such that, depending on the stimuli received, can remain quiescent or it can experience spiking activity. From the perspective of dynamical systems, several mathematical neuronal models have been suggested to study the bifurcations that govern this transition from resting to firing, and the equilibrium points, whether these are stable (resting or excitable state, see fig.1.1a and fig.1.1b) or unstable (sustained spiking, see fig.1.1c), as well as other aspects about its dynamics. One of the earliest proposals was made in 1907 by Louis Lapicque, with what is known as the *leaky integrate-and-fire*. Nowadays it is still used, mainly because of its simplicity, but there are other more biophysically accurate approaches such as the *Hodgkin-Huxley model* of the giant squid axon. Proposed in 1952 by Alan Hodgkin and Andrew Huxley, this model was pivotal in the understanding of the generation of action potentials in neurons, and became the reference for many models, which were built around it. For their work, they were awarded the 1963 Nobel Prize in Physiology or Medicine.

Having an understanding of single-cell dynamics, one can proceed to couple neurons to study their interaction in a network. A population of neurons can display interesting phenomena that is not inherent to an individual cell, like synchrony. For instance, some species of fireflies in a swarm are able to flash in a coordinated manner after some time[3], and increased synchrony has a role in some neurological disorders, such as epilepsy[13].

Other types of models called *firing-rate models* have been introduced to study the behaviour of a network. They describe properties of the population as a whole (the mean membrane potential, mean firing rate, etc.) rather than the particular state of each neuron, and they are widely used due to their simplicity, which allows for more theoretical results.

In the remainder of this chapter, we will give a brief overview of why neurons emit electric signals and present some of the ways one can model a single neuron.Then, we will review in more detail the model of choice for our work: the *quadratic integrate-and-fire* (QIF). Finally, we will see how we can model a network, with a high-dimensional system of coupled heterogeneous QIF neurons, and with a novel firing-rate model, both presented in [16].

1.1 What makes a neuron fire?

A neuron receives thousands of signals through the synapses. These inputs generate a current across the membrane which alters the membrane potential of this cell, generating what is called a *postsynaptic potential* (PSP). If the current is weak, we obtain small PSPs, and if it is strong, we have high PSPs. Furthermore, there are channels embedded in the membrane which, depending on the voltage, can amplify the PSPs.

The basic idea is that a neuron sums the PSPs from all the inputs and, with the additional amplification from the voltage-sensitive channels, it can surpass the firing threshold, and consequently generate an *action potential* (or *spike*), see fig.1.1b-c. These spikes are the main events that trigger the communication among neurons, and it is interesting to see how two identical neurons can have different responses to the same transmission and, conversely, how two different neurons can have the same response to an input.

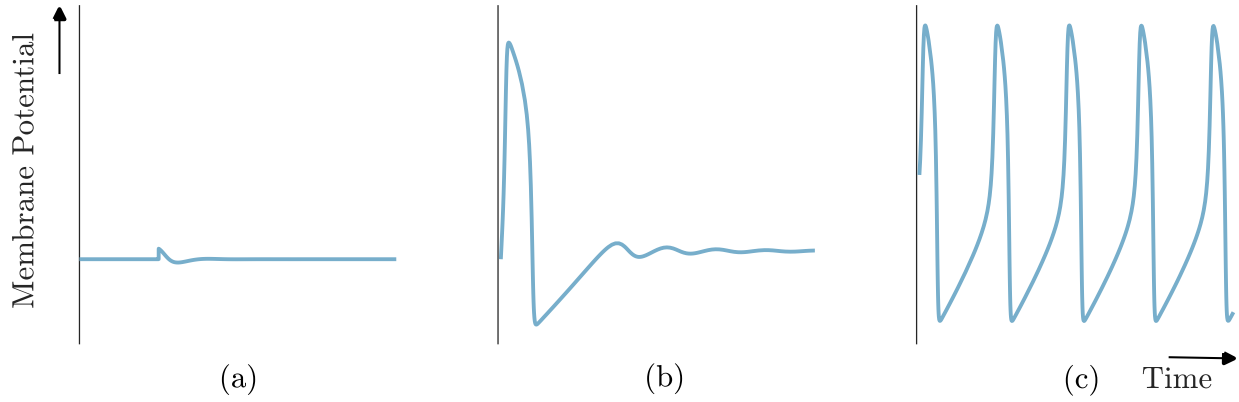

Figure 1.1: (a) Resting, (b) excitable and (c) sustained spiking states.

1.2 Modeling a single neuron

We have very precise neuronal models from the biological standpoint, such as the classic by Hodgkin and Huxley[9], for the particular case of a giant squid axon. However, as it takes into account the biophysical phenomena related to the generation of spikes, we have an intricate model, given by

$$
\begin{aligned}
C_m \frac{dV_m}{dt} &= \overline{g}_K n^4 (E_K - V_m) + \overline{g}_{Na} m^3 h (E_{Na} - V_m) + \overline{g}_l (E_l - V_m) + I, \\
\frac{dn}{dt} &= \alpha_n(V_m)(1-n) - \beta_n(V_m) n, \\
\frac{dm}{dt} &= \alpha_m(V_m)(1-m) - \beta_m(V_m) m, \\
\frac{dh}{dt} &= \alpha_h(V_m)(1-h) - \beta_h(V_m) h,
\end{aligned}
$$

where the α's and β's are functions obtained by interpolating points found experimentally, and are defined by

$$\alpha_n(V_m) = \frac{0.01(10 - V_m)}{\exp\left(\frac{10-V_m}{10}\right) - 1}, \qquad \beta_n(V_m) = 0.125 \exp\left(\frac{-V_m}{80}\right);$$

$$\alpha_m(V_m) = \frac{0.1(25 - V_m)}{\exp\left(\frac{25-V_m}{10}\right) - 1}, \qquad \beta_m(V_m) = 4 \exp\left(\frac{-V_m}{18}\right);$$

$$\alpha_h(V_m) = 0.07 \exp\left(\frac{-V_m}{20}\right), \qquad \beta_h(V_m) = \frac{1}{\exp\left(\frac{30-V_m}{10}\right) + 1}.$$

It is a system of nonlinear diferential equations in $\mathbb{R}^4$, which complicates the study of the phase space. Without getting much into it, functions n, m and h are involved in the regulation of the opening and closing of channel gates of the neuron, which let in and out of the cell the different types of ions that create a difference in electric potential across the membrane, measured by V_m.

We have other mathematical representations of a neuron, like the FitzHugh-Nagumo model[8][18] which is not as faithful to the biological interpretation, but preserves the qualitative features of the neurons' dynamics described in the H-H model, while at the same time reducing its dimensions to two. This means that we can study the dynamics in $\mathbb{R}^2$ and so, it helps interpreting and explaining some of the phenomena surmised from the Hodgkin-Huxley model. The system is

$$\varepsilon \frac{dv}{dt} = f(v) - w + I,$$
$$\frac{dw}{dt} = v - \gamma w,$$

where

$$f(v) = v(1 - v)(v - \alpha) \quad \text{for } 0 < \alpha < 1, \varepsilon \ll 1.$$

A fundamental property of neurons is *excitability*. As defined in [13], the textbook description of neuronal excitability is that a small stimulus will not generate a spike, whereas a large enough pulse will. This means we have a *threshold*. From the dynamical systems viewpoint, the subthreshold response translates into: all the trajectories that start close enough to the equilibrium will converge to it (this is the so-called *resting state*, see fig.1.1a). The supratheshold response translates into leaving a certain neighborhood of the equilibrium, resulting in a large-amplitude piece of trajectory, which then returns, and it either converges to the equilibrium (fig.1.1b) or continues with sustained oscillations (fig.1.1c). Therefore, what gives us information is not so much the shape of the spikes but their absence or presence. In fact, in 1948, Hodgkin identified the different types of responses, and suggested a classification of neurons according to their excitability[13]:

- *Class I*: The neuron can spike at arbitrarily low frequencies, depending on the strength of the current. The neurons that undergo a saddle-node bifurcation on invariant circle (SNIC) are the ones associated with this class, as the oscillations are born with infinite period and it gradually decreases as the intensity grows.

- *Class II*: The spiking frequency is bounded from below, and changing the strength of the intensity barely alters it. In general, the neurons in this class undergo a Hopf bifurcation, as the oscillations emerge with non-zero frequency.

- *Class III*: The neuron can exhibit a single spike in response to a pulse of current, and only with really strong injected currents, one can observe more than one spike in succession or none at all. Hodgkin referred to these as "sick neurons"; some books do not even consider this class, and focus on the first two.

A drawback of the aforementioned models is that, even though their parameters can be measured experimentally, these are usually an average from different cells[13], which can lead to the model exhibiting a different behaviour from the experiments. This is why we will consider *integrate-and-fire* (IF) models. The basic idea behind IF models is that when the membrane potential V reaches a certain peak value, the neuron is said to fire a spike, and the voltage is reset. Integrate-and-fire models provide a faithful reproduction of basic neurocomputational features related to excitability, such as the timing of the action potentials and how it is affected depending on different stimuli, which is the basic element in the communication between neurons in a network.

Furthermore, even though the previous models allow for a very detailed analysis of the behaviour of a single neuron, if we want to adapt them for the study of a neural network, it may lead to efficiency problems due to the complexity of the systems (not to mention that, depending on the parameters used, both H-H and F-N are stiff differential equations). This makes the computational treatment of networks more difficult, so that considering integrate-and-fire models can be helpful, as they are governed by only one first-order ODE (note that, in some cases, we can even obtain an exact solution, so they are easier to study analytically). Moreover, when working with networks, the intrinsic properties of each neuron are not as relevant, so working with these types of models makes more manageable the study of their properties as a collective.

An example of an IF model is the leaky integrate-and-fire (LIF, attributed to Louis Lapicque[14]) which, after rescaling, can be written as

$$\frac{dV}{dt} = b - V, \quad \text{if } V = 1 \text{ then } V \leftarrow 0.$$

The idea is what we described earlier: when the membrane potential V reaches a threshold (in this case $V_{thresh} = 1$), then the neuron is said to fire a spike, and the voltage is reset (here $V_{reset} = 0$).

1.2.1 Quadratic integrate-and-fire

For our work, we will use one of the simplest models of a spiking neuron, the quadratic integrate-and-fire (QIF, suggested by Izhikevich[11][12][13]). It can be obtained from a more general model called the *theta model* (introduced by Ermentrout and Kopell[6]), via a change of variables. The quadratic integrate-and-fire model is a class I neuron, and it is described by the following first order differential equation[1]

$$\frac{dV}{dt} = I + V^2, \quad \text{if } V = +\infty \text{ then } V \leftarrow V_{reset}, \tag{1.1}$$

where V is the membrane voltage variable, and I is the input current, which will usually be taken as a constant. Here, the voltage is reset when it reaches $+\infty$ (we will see that this happens in finite time). However, when doing simulations, we will fix a finite value V_{peak} (or V_{thresh} as in the LIF model) instead of $+\infty$ which will have the same role.

It is a Riccati's equation and it can also be solved by separation of variables. Either way, its analytical solution for $I > 0$ is

$$V(t) = \sqrt{I} \cdot \tan\left((t + C)\sqrt{I}\right),$$

for some constant C, which depends on the initial condition $V(t_0) = V_0$. Solving the initial value problem yields

$$V(t) = \sqrt{I} \tan\left(\arctan\left(\frac{V_0}{\sqrt{I}}\right) + \sqrt{I}(t - t_0)\right). \tag{1.2}$$

Notice that the voltage goes to infinity in a finite time; this is why we have the resetting mechanism described earlier. After rescaling, one can take $V_{peak} = 1$ (as seen in [13]) but in some instances it is useful to take $V_{peak} = +\infty$ for analytical results (see, for example, [7]).

For $I < 0$, we consider $J = -I$ and proceed with separation of variables, obtaining

$$V(t) = \frac{2\sqrt{-I}}{1 - e^{2\sqrt{-I}(t-t_0)}\left(1 - \frac{2\sqrt{-I}}{V_0 + \sqrt{-I}}\right)} - \sqrt{-I}. \tag{1.3}$$

Finally, for $I = 0$, the solution is

$$V(t) = \frac{V_0}{1 - V_0(t - t_0)}. \tag{1.4}$$

As a final note, we have the analytic solution, but if we wanted to solve it numerically, Matlab's `ode45` (or another ODE solver) can be used, together with an `events` function to check if V_{peak} has been reached. Also, all the step-by-step calculations can be found at the end of appendix A.

[1] This is the nondimensionalized version, which is its simplest form as well. For more details, see appendix A.

1.2.1.1 Dynamics of a QIF neuron

The next step is to run simulations to confirm and better understand some of the phenomena governed by (1.1).

Since the right-hand side of (1.1) is a quadratic function, we may have two, one or no roots, depending on the value of I, as it moves the parabola up and down. These roots determine the equilibria and, depending on V_0 and V_{reset}, different phenomena can be observed:

With negative input current I, $f(V) = I + V^2$ has two real roots, namely $V_\pm = \pm\sqrt{|I|}$. In the one-dimensional case, the stability is given by the slope of f, therefore, the negative root V_- is a stable point (the resting point) and the positive one is unstable (the threshold point V_+). As long as the initial value V_0 is below the threshold, the voltage will always converge to the resting point (see fig.1.2).

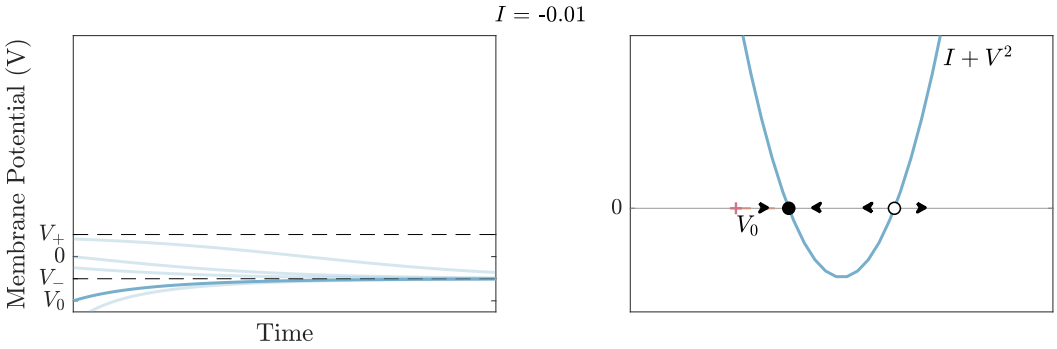

Figure 1.2: QIF resting state ($I < 0$, $V_0 < V_+$)[2]

Otherwise, if the initial value is above the threshold, the neuron will fire a spike, and the following response will depend on whether the reset value V_{reset} is sub or suprathreshold. If $V_{reset} < V_+$, a spike is fired, the voltage resets and then converges to the resting state, as seen in fig.1.3. But if $V_{reset} > V_+$, there is a first action potential, and then it starts the periodic spiking (see fig.1.4).

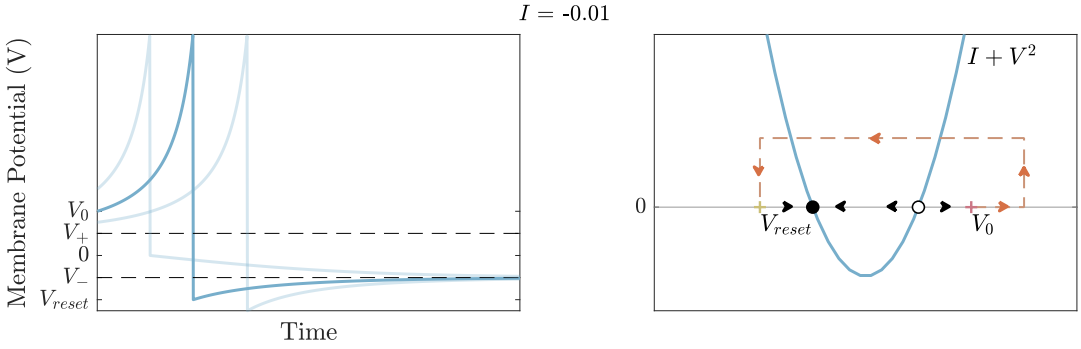

Figure 1.3: QIF excitable state ($I < 0$, $V_0 > V_+, V_{reset} < V_+$)

[2] From now on, in the figures, filled and empty circles represent stable and unstable equilibria respectively.

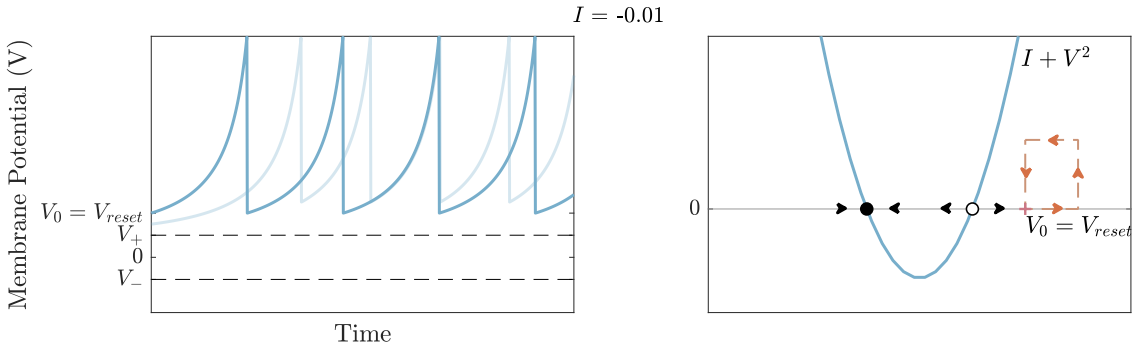

Figure 1.4: QIF bursting ($I < 0$, $V_0 > V_+, V_{reset} > V_+$)

This last scenario does not correspond to regular spiking since, after firing a spike, the neuron would need to recover, that is, the membrane potential should return to the resting voltage, which is smaller than V_+. However, it might make sense in the context of *bursting*[13], but this is another type of phenomena that we will no explore here.

When I reaches 0, the equilibrium points merge into one ($f(V) = V^2$ has double root 0) and originates a saddle-node (fold) bifurcation (see fig.1.5). The initial values to its left will make the membrane potential converge to equilibrium, and the ones to its right will be repelled. After the bifurcation, the equilibria have annihilated each other and no resting points exist. This means that whatever the initial and reset values are, the model will exhibit sustained spiking (fig.1.6).

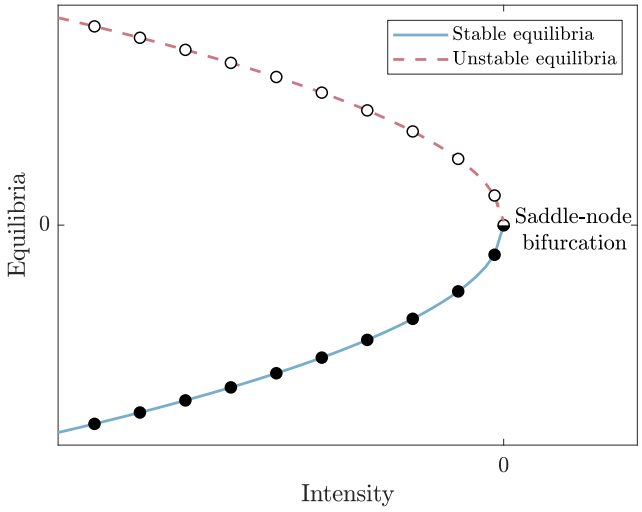

Figure 1.5: Equilibria of (1.1) as a function of I

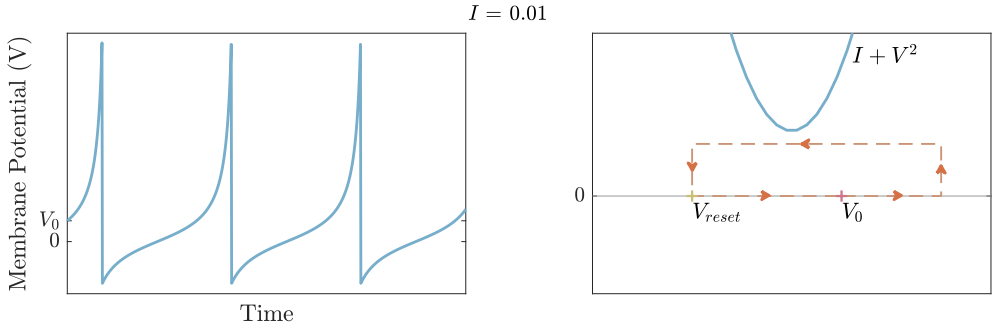

Figure 1.6: QIF periodic spiking ($I > 0$)

As we mentioned earlier, the quadratic integrate-and-fire models a class I neuron. However, this category of neurons are characterized by undergoing a saddle-node on invariant circle bifurcation, where the center manifold makes a homoclinic loop, and, as we have seen, $\dot{V} = I + V^2$ has a fold bifurcation when $I = 0$. But, if we add the resetting mechanism, it actually becomes a SNIC bifurcation. This might be easier to see if we change to the theta model: Taking $V = \tan(\theta/2)$, one obtains $\dot{\theta} = 1 - \cos\theta + (1 + \cos\theta)I$, with θ lying on the unit circle (when $\theta = \pi$, the neuron spikes). Now, looking at fig.1.7, we see that for negative currents we have a heteroclinic orbit connecting both fixed points. As I increases, they get closer, until they collide, forming a homoclinic orbit at $I = 0$. Actually, if we take solution (1.4) for $I = 0$, we see that it takes infinite time to close the orbit. Finally, for $I > 0$, we are left with a periodic orbit. This is the description of a SNIC bifurcation.

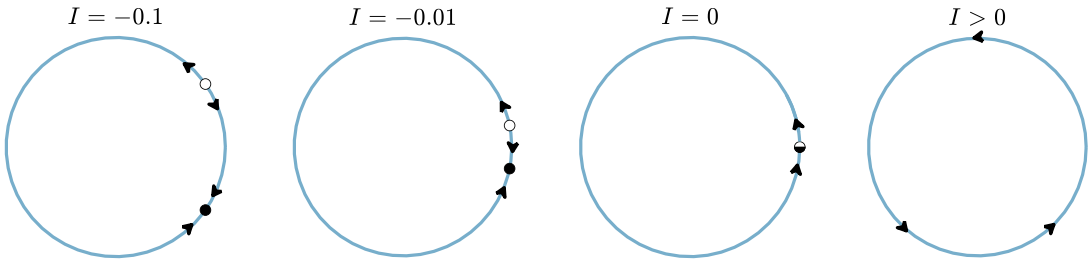

Figure 1.7: Saddle-node bifurcation on invariant circle in the theta model.

1.3 Modeling a network of neurons

When studying the behaviour of several neurons connected in a network, one might be interested in the dynamics of each individual cell. However, sometimes it is preferable to deal with models that capture the nature of the network as a whole, describing it in terms of macroscopic measures, such as the mean membrane potential or the firing rate (the mean rate at which neurons emit spikes). These macroscopic descriptions are usually called *firing-rate models* or *firing-rate equations* (FREs). The main advantage of FREs compared to the microscopic approach with large networks is their simplicity, which allows for mathematical analysis, and also, they are computationally efficient. Still, traditional firing-rate models such as the Wilson-Cowan equations[7] do not give a precise relationship between the microscopic dynamics of individual neurons and the macroscopic state of the network, and furthermore, do not describe settings where a fraction of the neurons are in synchrony[16].

In this project, we will work with the two network models presented in [16], where the authors derive a system of FREs given by

$$\dot{r} = \frac{\Delta}{\pi} + 2rv, \tag{1.5a}$$

$$\dot{v} = v^2 + \overline{\eta} + Jr + I(t) - \pi^2 r^2, \tag{1.5b}$$

with parameters Δ, $\overline{\eta}$, J and a function $I(t)$. The overdot denotes the derivative with respect to time. This system is for an all-to-all network of N heterogeneous QIF neurons, and its

microscopic model is given by the membrane potentials $\{V_j\}_{j=1,\dots,N}$, which are governed by

$$\dot{V}_j = I_j + V_j^2, \quad \text{if } V_j = V_{peak} \text{ then } V_j \leftarrow V_{reset}, \tag{1.6}$$

where the input current I_j has the form

$$I_j = \eta_j + Js(t) + I(t), \tag{1.7}$$

with the mean synaptic activation $s(t)$ (which is what links the neurons) written as

$$s(t) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k|t_j^k} \int_{-\infty}^{t} a_\tau(t - t')\delta(t' - t_j^k)\, dt'. \tag{1.8}$$

Here, t_j^k is the time of the kth spike of the jth neuron, $\delta(t)$ is the Dirac delta function, and $a_\tau(t)$ is the normalized synaptic activation caused by a single presynaptic spike with time scale τ. The parameters and functions will be described with more detail in the following sections, but for further explanations, we refer to [16].

Their results show the correlation between the spike generation mechanism of individual neurons (1.6), and the firing-rate (r) and mean membrane potential (v) coupling given by (1.5). In fact, this correspondence is exact in the thermodynamic limit (i.e. $N \to +\infty$).

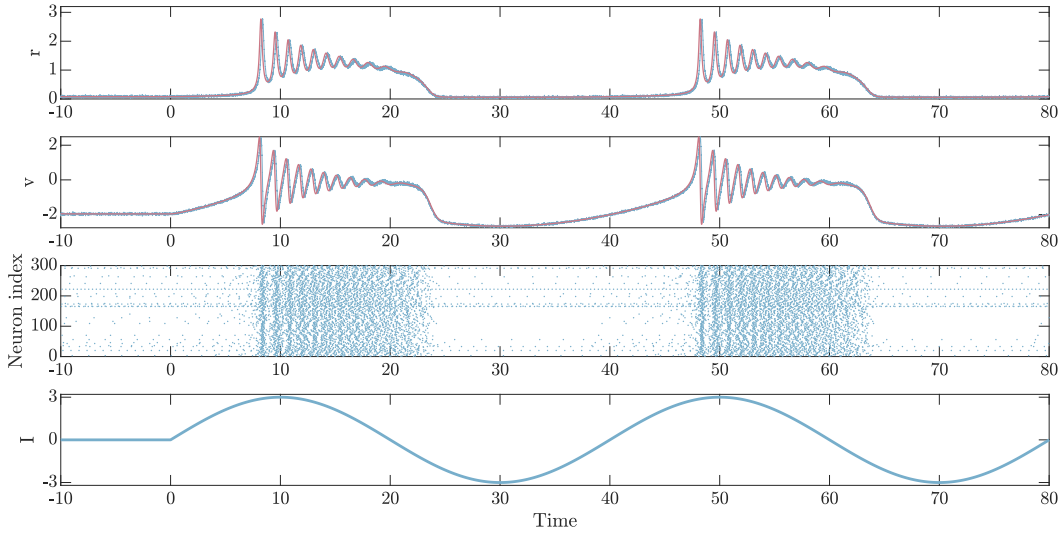

Figure 1.8: Simulations of both models (FREs in red and large-scale network in blue). Equations (1.5a)−(1.5b) describe exactly the model given by (1.6). Here $N = 10^4$, $J = 15$, $\overline{\eta} = -5$, $\Delta = 1$. $I(t) = I_0 \sin(\omega t)$ for $t \geq 0$ and $I(t) = 0$ otherwise ($I_0 = 3$, $\omega = \pi/20$) is shown in the last plot. Also drawn is a raster plot of 300 randomly selected neurons, which marks with a dot if the neuron j at time t fires a spike. To compute the firing rate, we count the number of spikes within $[t - \delta t, t]$ and divide it by N and δt ($\delta t = 2 \cdot 10^{-2}$). The mean membrane potential is computed considering only the population that is not in refractory state.

1.4 Outline of the project

In this project, our goal is to understand how the interaction between neurons can affect each individual one to the point where it may elicit a type of behaviour that it would not exhibit on its own. In this introductory chapter, we have provided the background necessary for the following sections.

In chapter 2, we will define the *phase response curve* (PRC), which is what will give us information about how a neuron responds to a perturbation. We will see that the quadratic integrate-and-fire actually has a closed form for the expression of its PRC, so we will derive it and study its properties.

In chapter 3, we will consider a population of coupled heterogeneous QIF neurons. We will explain with more detail equations (1.6)−(1.7)−(1.8) and we will run simulations of the network they describe. Then, we propose an algorithm to couple PRCs. We will discuss its implementation and compare the results that our method yields with both the QIF network and the mean-field model, to see if we are describing qualitatively the same activity of the population of neurons. This algorithm will help us understand how the different stimuli received from the network can push a neuron to spike earlier and, in particular, how knowing the shape of the PRC can help us achieve a synchronous state in the network.

In chapter 4, we give a summary of the conclusions we have reached and discuss future work.

2 Neuron's response to perturbations. The phase response curve

Given a neuron exhibiting periodic spiking, if it receives a brief stimulus that causes a change of its membrane potential, it can advance or postpone the next spike. Interestingly, this response may vary depending on when this stimulus takes place. In order to study these variations, we will consider the associated phase response curves (PRC), a well-known tool for the analysis of the interaction between neurons in a network.

2.1 Phase response curve

Let us consider a system with a limit cycle which, in our context, is a neuron displaying sustained spiking. We define *phase*, which will be denoted by θ, as the time elapsed since the last spike. Therefore, $\theta \in [0, T)$, where T is the period of oscillation.

Now, suppose, as in fig.2.1, that we inject a stimulus at phase θ (i.e., we increase by A units the variable that describes the membrane potential) and this pulse causes the neuron to fire earlier, and so, the phase is reset to a greater one θ_{new} (in short: $V(\theta) + A = V(\theta_{new})$). The map from θ to θ_{new} is called the *phase transition curve* (PTC) and the PRC for each phase is defined as

$$\mathrm{PRC}(\theta) = \mathrm{PTC}(\theta) - \theta = \theta_{new} - \theta.$$

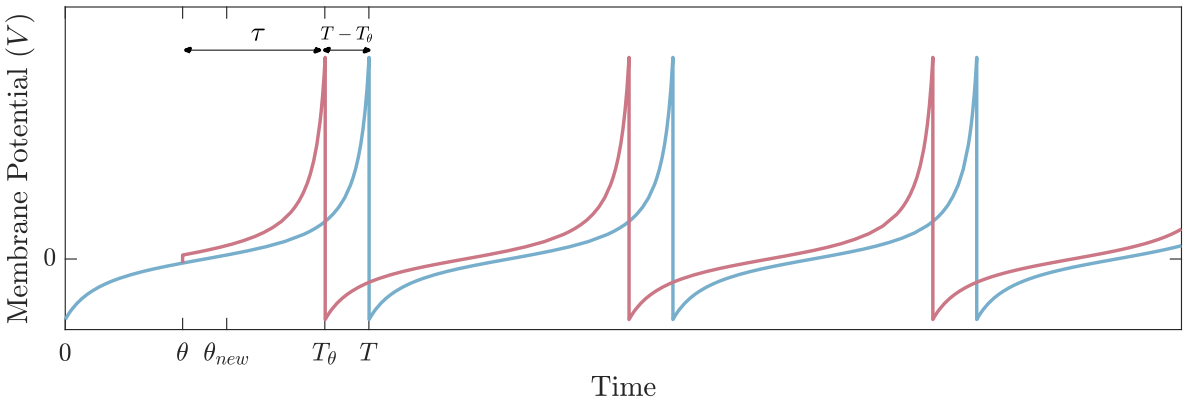

Figure 2.1: QIF unperturbed (blue) and after stimulation (red). In this case, the pulse at θ advances the action potentials. The PRC is the time shift between the old and new spikes.

Now, still in the same setting as in fig.2.1, we have the period T, which is the time between spikes for the unperturbed neuron. If there is a pulse at θ, we define τ as the time until the new spike. This is equivalent to writing $\tau = T - \theta_{new}$. Therefore, the time of the new spike is $T_\theta = \theta + \tau = \theta - \theta_{new} + T$, from which we deduce

$$\mathrm{PRC}(\theta) = T - T_\theta.$$

This expression will be useful in the following computations. We note that even though we call it "curve", the PRC is, of course, a map. However, what is usually presented is its graph.

2.1.1 PRC of the QIF model

The phase response curve (also called *phase-resetting curve*) is usually computed numerically, using the so-called *adjoint method*[7] (derived from *Malkin's theorem*[13]), but one of the advantages of having the analytical solution of the QIF (1.1), is that we can easily obtain the exact expression of its PRC:

For $I > 0$, which is when we have periodic spiking, we will consider $V_0 = V_{reset}$, so we can work in the interval $[t_0, t_f]$, where t_f is the time when the first spike is fired (this way, we have slightly nicer expressions). By definition, $V(t_f) = V_{peak}$, therefore, using solution (1.2) we have

$$V(t_f) = V_{peak} \iff \sqrt{I}\tan\left(\arctan\left(\frac{V_{reset}}{\sqrt{I}}\right) + \sqrt{I}(t_f - t_0)\right) = V_{peak}$$

$$\implies t_f = \frac{1}{\sqrt{I}}\left(\arctan\left(\frac{V_{peak}}{\sqrt{I}}\right) - \arctan\left(\frac{V_{reset}}{\sqrt{I}}\right)\right) + t_0, \tag{2.1}$$

and if we now take $t_0 = 0$, then t_f is the period T.

To compute T_θ for some $\theta \in [0, T]$, we suppose there is a brief pulse at $t = \theta$ that raises the membrane potential by A units, and then we resume the integration from $(t, V) = (\theta, V(\theta) + A)$. Then, similar to what we have just seen for T, we obtain

$$T_\theta = \frac{1}{\sqrt{I}}\left(\arctan\left(\frac{V_{peak}}{\sqrt{I}}\right) - \arctan\left(\frac{V(\theta) + A}{\sqrt{I}}\right)\right) + \theta, \tag{2.2}$$

and using (1.2) to evaluate $V(\theta)$, we end up with

$$T_\theta = \frac{1}{\sqrt{I}}\left(\arctan\left(\frac{V_{peak}}{\sqrt{I}}\right) - \arctan\left(\frac{A}{\sqrt{I}} + \tan\left(\arctan\left(\frac{V_{reset}}{\sqrt{I}}\right) + \sqrt{I}\theta\right)\right)\right) + \theta \tag{2.3}$$

and so

$$\mathrm{PRC}(\theta, A) = \min\left\{\frac{1}{\sqrt{I}}\left(\arctan\left(\frac{A}{\sqrt{I}} + \tan\left(\arctan\left(\frac{V_{reset}}{\sqrt{I}}\right) + \sqrt{I}\theta\right)\right) - \arctan\left(\frac{V_{reset}}{\sqrt{I}}\right)\right), T\right\} - \theta \tag{2.4}$$

The reason we write the minimum is because for every A, there is a point θ^* where this perturbation is just enough to reach V_{peak} so, for any $\theta \in [\theta^*, T]$, it turns out that $T_\theta = \theta$; as a consequence, $\mathrm{PRC}(\theta, A) = T - \theta$ and, on that interval, the curve is a straight line with

slope -1. In fact, in eq.(2.2), if $V(\theta) + A > V_{peak}$, because $\arctan(x)$ is a strictly increasing function, we would have $T_\theta < \theta$, and this is not coherent if T_θ is the time of the new spike after the perturbation at phase θ.

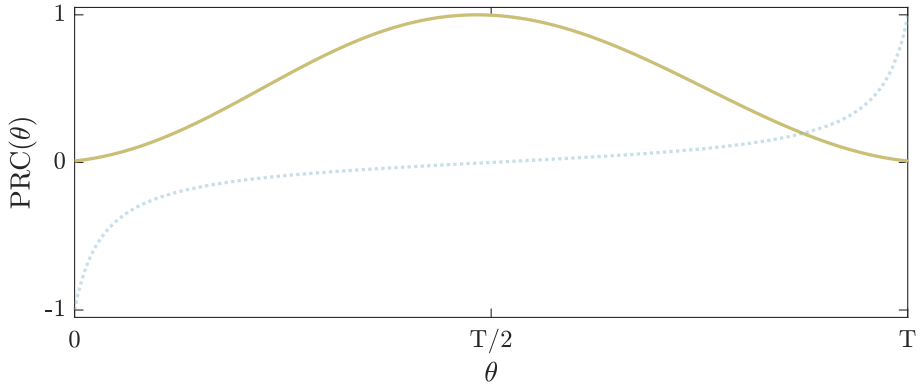

Figure 2.2: PRC of (1.1) with $I = 0.01$, $A = 0.01$ and $V_{peak} = -V_{reset} = 1$. PRC($\theta$) has been rescaled, so it is easier to compare it against the unperturbed $V(t)$ (dotted line). Bear in mind that the PRC has units of time and V is a voltage, so it is not rigorous to compare them in quantitative terms, but it is interesting to explain the shape of the PRC using $V(t)$.

An important observation to make regarding fig.2.2, is that the PRC is always positive, which means that the spikes are always advanced in time whenever there is a positive voltage increase A after a stimulus (if $A > 0$ we say that the stimulus is excitatory). If $A < 0$ (i.e. the stimulus is inhibitory), we would have a negative PRC and the spikes would be delayed. Other models might exhibit delay or advancement depending on the phase of stimulation (for example, both Hodgkin-Huxley and FitzHugh-Nagumo models have a sinusoidal form[2]).

Notice as well that the PRC displayed in fig.2.2 looks (somewhat) symmetrical. This is in part due to the symmetrical shape of the solution, but this only happens for small stimuli. If we compute the PRC for increasing strengths of the perturbation, we will see a gradual tilt of the curve to the left (see fig.2.3).

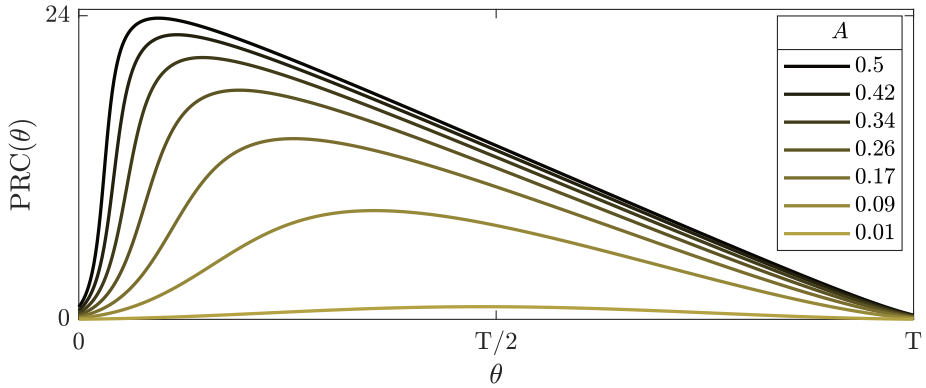

Figure 2.3: Comparison of PRCs for the QIF model (1.1) with $I = 0.01$ for increasing values of the perturbation parameter $A > 0$ (without scaling).

To explain this shift, if we go back to fig.2.2, we see that there is an interval in the middle, where the membrane potential increases very slowly . If, before we reach that interval, there is a large enough stimulus that skips it, then the phase advances significantly. The perturbation will be most effective at the point where the PRC reaches its maximum, therefore, the greater the impulse, the earlier we can jump over this region, and this is why the peak of the curve shifts to the left.

In fact, this can also be deduced from the ODE itself, because the interval where the dynamics are slower is around $V = 0$, as it is the point where the slope of the solution is the smallest (see fig.1.6).

Finally, if we consider the PRC as a function of the two variables θ and A, we obtain the plot in fig.2.4. We see what we have just discussed, which is that the higher the impulse, the more prominent will be the tilt to the left (and, of course, the greater will be the advance of the phase). We can do the same for the PTC. If there is no perturbation ($A = 0$), the phase will stay the same, so the PTC is the identity function. And since $\mathrm{PTC}(\theta, A) = PRC(\theta, A) + \theta$, what we have just observed, applies here as well: the greater the impulse, the greater the slope and also, the region where this slope is greater, will be closer to the left (closer to $\theta = 0$).

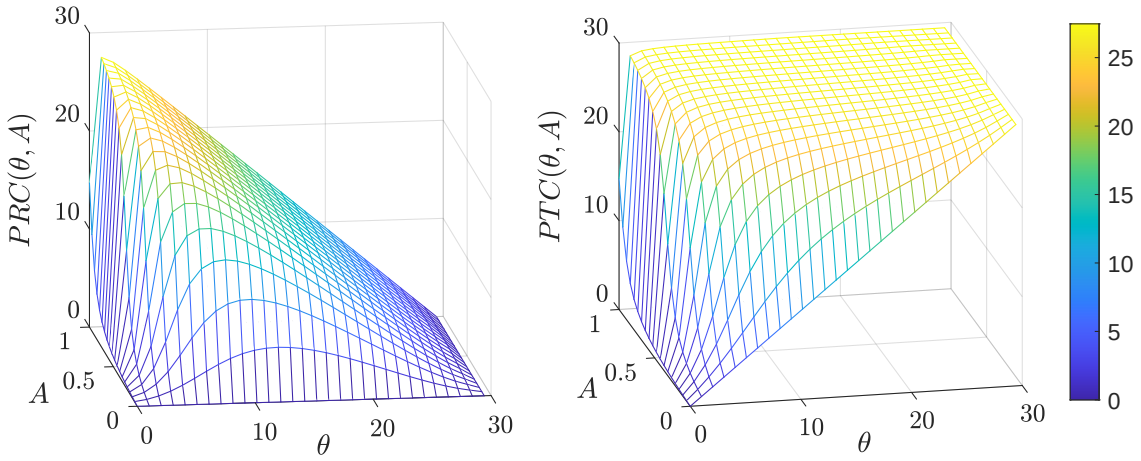

Figure 2.4: PRC and PTC plots considered as functions of the two variables θ and A.

3 All-to-all coupling of PRCs

In this chapter we will describe the microscopic model for a network of N heterogeneous quadratic integrate-and-fire neurons connected all-to-all. Then, we will proceed to study the coupling of PRCs. To do that, we propose an algorithm that reflects the dynamics of the network, similarly to what we see in the microscopic network, and will help us understand how the transmission of electric impulses affects each neuron (in terms of its phase) and the network as a whole.

3.1 Network of heterogeneous QIF neurons

In section 1.3, we have already presented an overview of how we can model a network of QIF neurons connected all-to-all. In this section we explain with more detail the role of some of the parameters and functions, as it will help us later in the construction of the algorithm to couple PRCs, and it may give us hints about what to expect when we change the parameters in the simulations.

3.1.1 Model description

As we anticipated in section 1.3, the membrane potential of each neuron j follows the ODE

$$\dot{V}_j = I_j + V_j^2, \quad \text{if } V_j = V_{peak} \text{ then } V_j \leftarrow V_{reset}, \tag{3.1}$$

where the input current I_j is the sum of three distinct components:

$$I_j = \eta_j + Js(t) + I(t). \tag{3.2}$$

The term η_j represents a constant input current different for each neuron, and it is used to introduce heterogeneity. In [16], the η_j's are chosen so that they follow a Lorentzian (or Cauchy) distribution, but in the same paper there is a comparison using different distributions, and the results are shown to be qualitatively similar. The function $s(t)$ is the *mean synaptic activation*, and it describes the synaptic activity according to the number of spikes in the network at times t' such that $t' \leq t$ (it can be expressed as (1.8)). The parameter J is the *synaptic* (or *coupling*) *strength*. Since $s \geq 0$, having $J > 0$ means the coupling is excitatory (i.e. the inputs from other neurons increase the membrane potential), whereas $J < 0$ translates to having inhibitory coupling (the inputs from other neurons decrease the membrane potential). For $J = 0$, the network is uncoupled (the neurons are disconnected).

The greater $|J|$ is, the stronger will be the effect of the pulses from the other neurons. Therefore, J together with $s(t)$ model the current that each neuron is receiving as a function of the aggregate of spikes from all the neurons at that instant. Finally, $I(t)$ represents a common time-dependent external current.

3.1.2 Numerical simulations

To run simulations of a network of N neurons, we will use Euler's method to integrate each equation, with time step $dt = 10^{-4}$. It is the method of choice in [16], probably because the step size can be controlled, which determines the number of iterations and facilitates the implementation of the resetting mechanism and the refractory state of each neuron. Also, the precision obtained by using a small time step with Euler's method is enough for their simulations, but in some instances, it is necessary to improve the accuracy with higher-order methods. In fact, rather than improving the method of integration, it might be more important for the precision of the network to refine the spike time, as seen in [19], where the authors discuss a scheme that uses a second-order Runge-Kutta together with an interpolant to find more accurately the time an integrate-and-fire neuron reaches the threshold value.

We will take $V_{peak} = -V_{reset} = 100$, and the initial values for each V_j can be randomly chosen following a uniform distribution on the interval $[-100, 100]$.

As previously said, η_j follows a Lorentzian distribution. If Δ and $\overline{\eta}$ are its half-width and center, respectively, then one way to prescribe a probability distribution to a set of points is by using its quantile function, which is the inverse of its cumulative distribution function. In this case, the quantile function for a Lorentzian distribution is

$$Q(p; \overline{\eta}, \Delta) = \overline{\eta} + \Delta \tan\left(\pi\left(p - \frac{1}{2}\right)\right), \tag{3.3}$$

and so, taking $p = \dfrac{j}{N+1}$, η_j can be computed deterministically using the following expression (a plot of the distribution can be seen in fig.3.6):

$$\eta_j = \overline{\eta} + \Delta \tan\left(\frac{\pi}{2} \cdot \frac{2j - N - 1}{N+1}\right) \qquad j = 1, \ldots, N. \tag{3.4}$$

According to [16], the time it takes to reach $+\infty$ from V_{peak} is approximately $1/V_{peak} = 10^{-2}$, and similarly, it takes roughly $-1/V_{reset} = 10^{-2}$ to go from $-\infty$ to V_{reset}. So, following equation (3.1), once the membrane potential verifies $V_j \geq V_{peak}$ (we cannot check the equality, as we have discrete steps), we reset the voltage and the neuron enters what is called *refractory state*, which is the interval of time where the neuron produces the spike and recovers. During this time, the cell cannot generate more spikes, regardless of the external inputs, as it is still generating the initial one. After 10^{-2} units of time (100 time steps in our case) have passed, we say that neuron j has fired a spike, and after 10^{-2} more, the refractory time finishes and the integration for V_j is resumed.

As we have seen previously, the mean synaptic activation function can be expressed as

$$s(t) = \frac{1}{N}\sum_{j=1}^{N}\sum_{k|t_j^k}\int_{-\infty}^{t} a_\tau(t-t')\delta(t'-t_j^k)\,dt'. \tag{3.5}$$

In the numerical simulations conducted in [16], a_τ is chosen to be

$$a_\tau(t) = \frac{\Theta(\tau - t)}{\tau},$$

where Θ is the Heaviside step function (zero for negative values and one for positive values) and $\tau = 10^{-3}$. So, expression (3.5) becomes

$$s(t) = \frac{1}{\tau N}\sum_{j=1}^{N}\sum_{k|t_j^k}\int_{t-\tau}^{t} \delta(t'-t_j^k)\,dt'. \tag{3.6}$$

We recall that t_j^k is the time of the kth spike of the jth neuron and $\delta(t)$ is the Dirac delta function. Then, equation (3.6) simply counts the number of spikes in the network within the time interval $[t-\tau, t]$ and divides it by N and τ to have the average in that period.

As a final note, for the following simulations, unless otherwise stated, we will use the QIF equation

$$\tau_m \dot{V} = I + V^2,$$

with $\tau_m = 10$ ms (taken from [15]). We are just rescaling the time, so the dynamics of the neurons are slowed down but they are qualitatively the same; this way, the following raster plots are much clearer. For more details, see appendix A (eq.(A.2)).

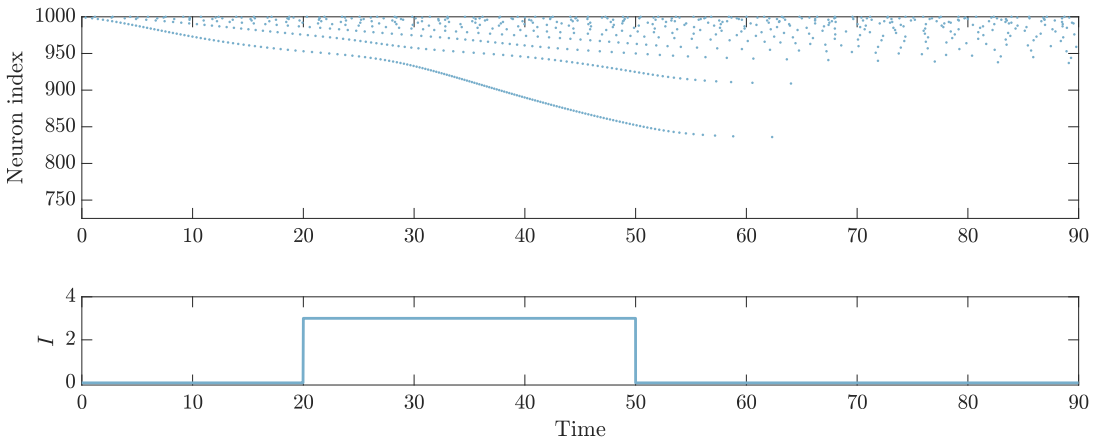

Figure 3.1: Raster plot corresponding to a network of $N = 1000$ neurons, with parameters: $\overline{\eta} = -5$, $\Delta = 1$, $J = 7$. At $t = 20$ a constant current $I = 3$ is applied to all the network, which pushes some of the neurons that were not in the oscillatory regime to now fire periodically. At $t = 50$ the external current is removed, and the neurons that had previously changed their dynamics, gradually stop spiking, and the network returns to its initial state. Neurons from 1 to around 750 are not shown, as they do not have high enough current to fire a spike. All the voltages are initialized to V_{reset}.

3.2 Coupling PRCs

After the description of the previous model for a QIF network and its numerical implementation, we propose an algorithm to study the interaction between neurons through their PRCs.

Following the idea of the theta model, instead of describing the evolution of an oscillating QIF in terms of its membrane potential, we can do it through its phase. For an uncoupled neuron, we have computed its period T, and so, we know that it will spike every time its phase reaches this value ($\theta \in [0, T]$). Then, when the neuron is connected to a network, we know how a perturbation provoked by another neuron's spike can advance or delay the phase, since we have the PRC. This is the main idea behind the algorithm.

Now we will proceed with the description of the method and its implementation. Then we will see how well it describes the population of neurons, by comparing it with the QIF network and the FREs and, finally, we will see what results we can obtain with it.

3.2.1 Algorithm

The scheme has three main steps, which are summarized as follows:

1. At time t, find the neuron k closest to firing, and register (k, t_k) (where t_k is the time it takes to fire). For every spike, add the pair (k, t_k) to the set (or list) S, and also, create a list L_t to mark the neurons that have spiked in the current iteration.

2. Advance all the phases t_k units, reset θ_k (as it has just spiked) and apply at each neuron the perturbation generated by this last spike (We will shortly see what we consider as the pulse A): $\forall j \neq k: \ \theta_j \leftarrow \theta_j + t_k + \mathrm{PRC}(\theta_j, A)$; $\theta_k \leftarrow 0$.

3. Check if this perturbation has caused any neurons to spike and, if so, register it (add it to S and L_t) and reset them. Add these new neurons that have spiked to a new list L. Apply the perturbations resulting from these new spikes to the neurons that do not belong to L_t. Empty the list L and add the new neurons that have spiked (if any). Repeat until L is empty, and then go to step 1 with $t \leftarrow t + t_k$.

The full algorithm is described in alg.1. Also, a Matlab implementation can be found in appendix C.

Algorithm 1 All-to-all coupling of PRCs

$\forall j:$ initialize θ_j (normalized to lie within $[0, 2\pi]$)
$\forall j:\ T_j \leftarrow$ period of neuron j
$S \leftarrow \varnothing$
$t \leftarrow t_0$
while $t \leq t_{end}$ **do**
  $\forall j:\ t_j \leftarrow$ time until the next spike
  $k \leftarrow \underset{j}{\operatorname{argmin}}\, t_j$
  $S \leftarrow S \cup \{(k, t_k)\}$
  $\theta_k \leftarrow 0$ (reset)
  $A \leftarrow Js(t) = J\dfrac{1}{N}$
  $\forall j \neq k:\ \theta_j \leftarrow \theta_j + \dfrac{t_k}{T_j}2\pi + \mathrm{PRC}(\theta_j, A)$
  $L \leftarrow k$
  $L_t \leftarrow k$
  while $L \neq \varnothing$ **do**
    $L \leftarrow \varnothing$
    $\forall j:$ **if** $\theta_j \geq 2\pi$ **then**
      $\theta_j \leftarrow 0$ (reset)
      $L \leftarrow L \cup \{j\}$
      $S \leftarrow S \cup \{(j, t_k)\}$
    end if
    $L_t \leftarrow L_t \cup L$
    $A \leftarrow Js(t) = J\dfrac{|L|}{N}$
    $\forall j \notin L_t:\ \theta_j \leftarrow \theta_j + \mathrm{PRC}(\theta_j, A)$
  end while
  $t \leftarrow t + t_k$
end while

Now, some remarks are in order:

- The phase of each neuron is normalized to lie between 0 and 2π. This rescaling makes things easier when comparing PRCs of neurons with different oscillation frequencies, since they now have the same domain and range. This means that if we use (2.4) to compute $\mathrm{PRC}(\theta, A)$, we first have to "denormalize" θ and then normalize the result if we want to update the phase.

- Of course, when we talk about the phase $\forall j$, we are referring to the neurons that are in the oscillatory regime, as it would not make sense to talk about the period of a

neuron that is at rest. Nevertheless, as we have previously seen in section 1.2.1.1, a high enough perturbation (or several small ones in a short amount of time) can evoke a spike from a neuron with $I_j \leq 0$. We are interested in registering these spikes, as they affect the whole network, so in parallel to alg.1, we will integrate the neurons with $I_j \leq 0$. We have the analytical solution, so we can easily control the evolution of these neurons at each time step t_k. However, equation (3.1) might not be the same as equation (1.1), because $I(t)$ can be a time-dependent function, as we saw in fig.1.8. This brings us to the next point:

- The introduction of an intensity that varies over time means that the ODE in (1.1) can change drastically, and the solutions and expressions we have found for the PRC and the period, may become invalid. For example, if we inject a sinusoidal current as in fig.1.8, some neurons would be constantly transitioning from resting to firing spikes periodically, and the PRC would change every instant. In the implementation for the QIF network described in section 3.1.2, at each integration step, each equation is integrated regardless of the expression of I_j, but our algorithm cannot keep track of these changes. However, if we consider a step function such as the one in fig.3.1 (also seen in [5][16]), we have a constant stimulus at a certain interval, so we only have to be careful at the points where this external input activates and deactivates. Namely, the period of each neuron will need to be recomputed, and so, each phase needs to be renormalized. With the increase of the current, some neurons will start spiking periodically, which means that their phase needs to be computed (we know how to compute the period, and similarly, we can compute the time to reach V_{peak} starting at a certain V_0, so the phase is the difference between both values). When it deactivates, we need to keep track, once again, of the voltage for the neurons that have just left the oscillatory regime (we can compute $V(\theta_j)$ with initial condition V_{reset} right before it deactivates). We also note that at the time step where the external current is turned on or off, we will first need to advance the phases up until the point of activation or deactivation, and then recompute t_k considering the new I_j. Other than at those two points, the algorithm proceeds as usual.

- As a final remark, we know exactly when each neuron spikes, so $s(t)$ does not need to compute the average in a certain time window, and we can consider the effect of the spikes at each time. This means that $s(t) = \frac{\text{number of spikes at time t}}{N}$. However, at every iteration in the inner loop, $s(t)$ only considers the new spikes, as the effect of previous ones has already been registered.

  Also, an important observation about the previous point and the algorithm, is that we are not considering $Js(t)$ as another term in the input current I_j, but as the perturbation A. This raises the question of whether we can use as a voltage pulse something that was considered as a current. The answer is *yes*, and we can prove it:

  For simplicity, we suppose that there is only one spike at $t = t_s$. Therefore, we can write s as $s(t) = \frac{1}{N}\delta(t - t_s)$ (δ is the Dirac delta function), and so, each neuron follows the equation

$$\dot{V} = V^2 + I + \frac{J}{N}\delta(t - t_s). \tag{3.7}$$

Now, we can follow the proof found in [4]: We can write the difference of the membrane potential right before and after the perturbation as $V(t_s) - \lim_{h\to 0^+} V(t_s - h)$. Then, writing $V(t)$ in its integral form to recover (3.7), we have

$$\begin{aligned} V(t_s) - \lim_{h\to 0^+} V(t_s - h) &= \int_0^{t_s} \dot{V}(t)\,dt + V(0) - \left(\lim_{h\to 0^+} \int_0^{t_s - h} \dot{V}(t)\,dt + V(0) \right) \\ &= \lim_{h\to 0^+} \int_{t_s - h}^{t_s} \dot{V}(t)\,dt = \lim_{h\to 0^+} \int_{t_s - h}^{t_s} \left(V^2(t) + I + \frac{J}{N}\delta(t - t_s) \right) dt \\ &= \frac{J}{N} + \lim_{h\to 0^+} \int_{t_s - h}^{t_s} \left(V^2(t) + I \right) dt = \frac{J}{N} \end{aligned}$$

and this ends the proof. We note that this argument is also valid if we replace $V^2 + I$ with a general function $f(V)$, as seen in the original proof.

3.2.2 Results and discussion

3.2.2.1 Comparison with the QIF network and the FREs

Now we will compare the previous scheme with the QIF network and the FREs. We are particularly interested in the differences we may have by not considering the refractory period for the neurons that have just fired, and how not having a fixed time step can affect the overall performance of the algorithm.

As the set S saves the time when each neuron fires, it is easy to generate a raster plot like the one in fig.3.1 for the network of QIFs. This way, we can (to a certain extent) validate our algorithm. Using the same parameters: $N = 1000$, $\overline{\eta} = -5$, $\Delta = 1$, $J = 7$, and the intensity $I(t)$ defined as

$$I(t) = \begin{cases} 3 & \text{if } t \in [20, 60] \\ 0 & \text{otherwise} \end{cases}$$

we obtain, qualitatively, the same dynamics, as seen in fig.3.2.

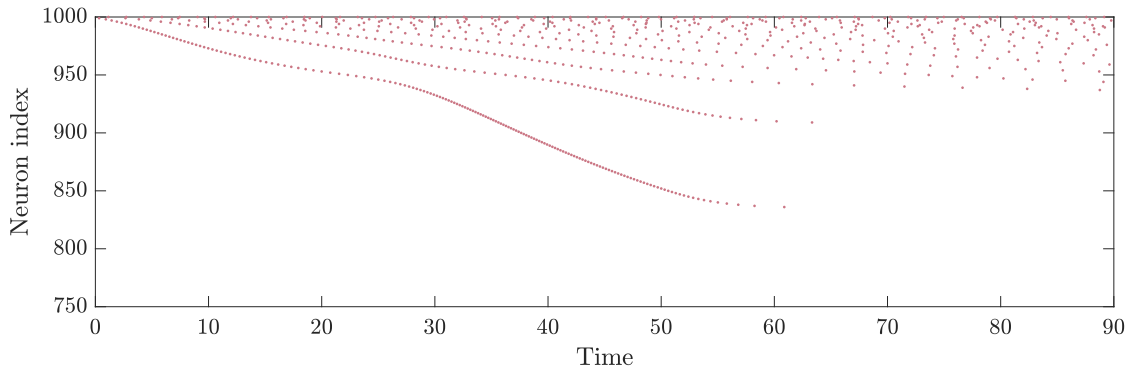

Figure 3.2: Raster plot obtained using alg.1, with the same parameters as in fig.3.1.

With a small modification to the code, we can also compute the mean firing rate and the mean membrane potential, and compare our results with the FREs, just like we did in section

1.3. For the mean firing rate, the set S already registers the spikes, so we count the number of spikes within $[t - \delta t, t]$ and divide it by N and $\delta t = 2 \cdot 10^{-2}$. For the mean membrane potential, we are already controlling the voltage for the neurons that are not periodically spiking. For the rest, we know their phase θ at each time step, therefore, using solution (1.2), we obtain their membrane potential by computing $V(\theta)$ with initial condition $V_0 = V_{reset}$. With that, we can compute the average for the entire population (we are not considering the refractory period).

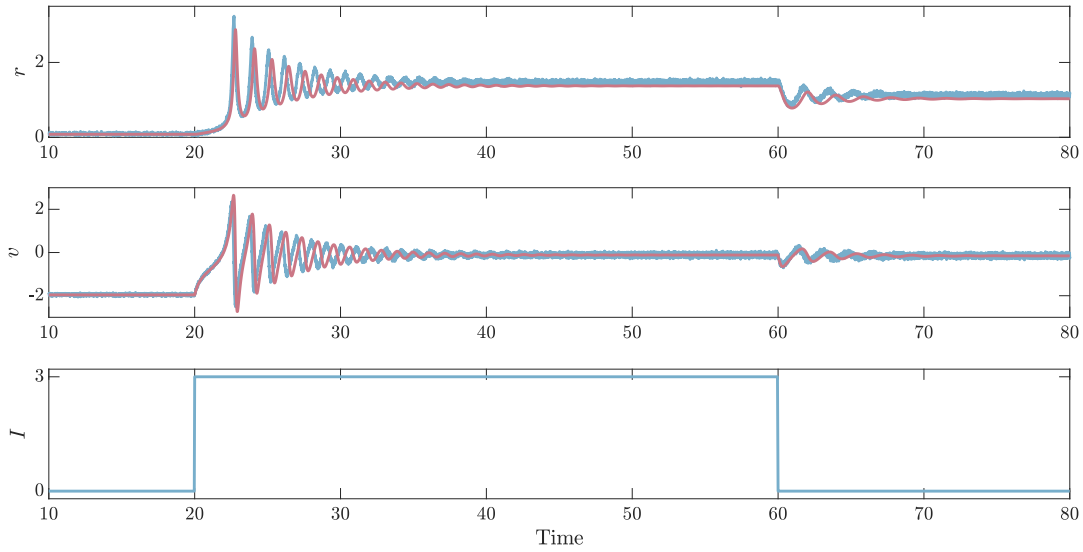

Figure 3.3: Comparison between the FREs (red) and our "network of PRCs" (blue). Here we have used the original QIF equation (1.1).

In fig.3.3, we notice that the PRC network has both r and v slightly shifted to the left, and the mean firing rate is higher in general, compared to the FREs. This is because we do not put the neurons in refractory state. This means that once they reach V_{peak}, they immediately fire a spike instead of waiting 10^{-2} ms, and they can keep firing without waiting 10^{-2} ms more, thus the increase of the firing rate. In this case, the simulation is done with $N = 10000$ neurons, as the FREs are exact in the thermodynamic limit, so 1000 neurons are not enough to do the comparison. This leads to a very slow execution of the algorithm compared to the QIF network. This is not so much because of efficiency issues with the tenfold increase of the dimension of the system, but because of how the algorithm is designed: for the QIF network, we have a fixed time step of 10^{-4} ms, while in our case every time step is the time it takes for the next neuron to spike, so it is not fixed. On the other hand, the shape of the quantile function (3.3) is given by a tangent function so, in our case, the greater the j, the greater η_j is, and therefore, the smaller is its period. So neuron $j = N$ has the shortest period, and for $N = 10^4$ it is approximately $T_N = 0.0375$ (for comparison, with $N = 10^3$ it is 0.1576). So, at most, the time step is T_N but in a network, all the phases keep advancing, and if there is high activity (and on top of that, $I(t)$ increases the overall intensity), the time step at some instances is reduced drastically. For reference, in fig.3.3, the mean time step is around 10^{-5}, but it goes as low as 10^{-11}. On the other hand, if the activity of the population is low, the time steps increase, and the runtime can be much lower.

3.2.2.2 When is a perturbation enough to elicit an immediate spike?

The advantage of our algorithm is that we can study the effect of an action potential on the rest of the neurons in the network. In particular, we may be interested in knowing if a neuron firing a spike can push another neuron to spike. Of course, if we fix the synaptic strength J to a really high value, we can force immediate spikes after a perturbation. However, we want to study scenarios that are as close as possible to the real dynamics of a biological network, so we need to choose reasonable parameters that help us obtain the response that we want from the system. We have to choose J, $\overline{\eta}$ and Δ so that the population manifests high activity. In this state, numerous perturbations will push the neurons much closer to firing, and this may lead to higher perturbations due to some neurons occasionally synchronizing. To find these parameters, we can use the firing-rate system of equations (1.5). As we mentioned in section 1.3, the simplicity of the FREs allows for an easier theoretical analysis of the network. With them, we obtain the stability diagram in fig.3.4. The region where there is a single stable node determines the parameters for which the population is in a state of low activity, because the center of the Lorentzian distribution is very low and the synaptic strength is upper bounded, so most neurons will generally be at rest, and the strength of the perturbations might not be high enough to push a neuron past the threshold, so the mean membrane potential and firing rate will converge directly to the stable node. We are interested in the region where the fixed point is a stable focus. This is where the network exhibits high activity, as for most of the values of $\overline{\eta}$, the majority of neurons are in the oscillatory regime, and for the ones that are not, J is usually high enough to force the neurons to spike. So, we will chose the parameters so that they fall on this region.

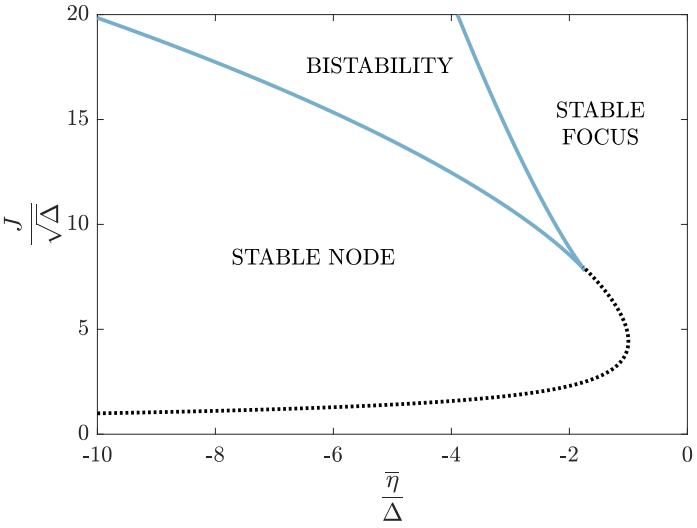

Figure 3.4: Stability diagram for the fixed points of system (1.5). The solid lines mark the points where a saddle-node bifurcation takes place. An exact parametric form can be obtained: $\left(\frac{\overline{\eta}}{\Delta}, \frac{J}{\sqrt{\Delta}}\right) = \left(-(\pi\tilde{r})^2 - \frac{3}{(2\pi\tilde{r})^2}, 2\pi^2\tilde{r} + \frac{1}{2\pi^2\tilde{r}^3}\right)$. The dotted line delimits the transition from stable node to stable focus, and its parametrization is: $\left(\frac{\overline{\eta}}{\Delta}, \frac{J}{\sqrt{\Delta}}\right) = \left(-(\pi\tilde{r})^2 - \frac{1}{(2\pi\tilde{r})^2}, 2\pi^2\tilde{r}\right)$. For the derivation of these expressions, see appendix B and for more details on the stability of the equilibria, see [16].

Now, as we have said, we want to check if a neuron (or several) firing, can immediately force another neuron to spike. The inner loop in alg.1 registers exactly that.

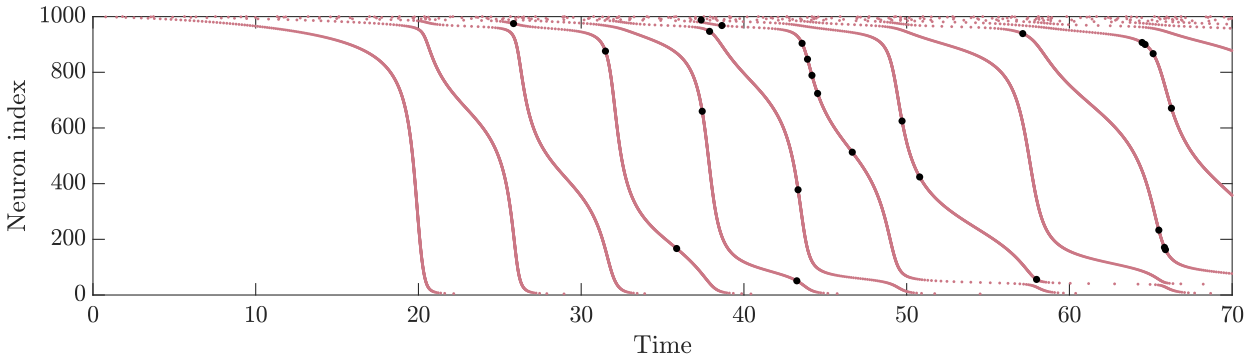

Figure 3.5: Raster plot using parameters: $\overline{\eta} = -0.5$, $J = 20$, $\Delta = 0.7$. From $t = 20$ to $t = 50$, $I(t) = 9$, and is it zero otherwise. We mark with a black dot the neurons that spike due to an impulse received at the same time. All the neurons are initialized at the reset value.

In fig.3.5, we notice that right after the external input $I(t)$ is added ($t = 20$), most of the neurons almost synchronize. This can be explained looking at the shape of the quantile function (3.3) in fig.3.6.

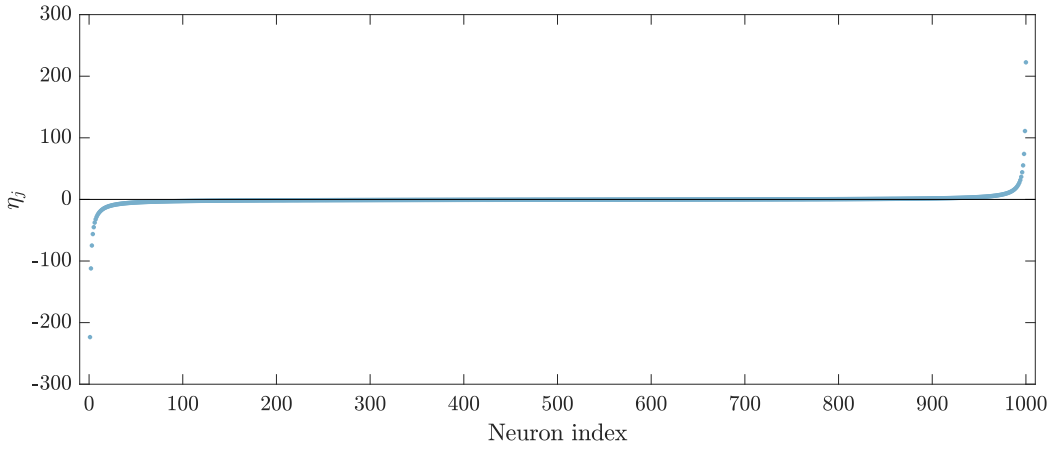

Figure 3.6: Intrinsic current for each neuron j when $I(t) = 0$, using the same parameters as in fig.3.5. For $0 < j < 699$, we have $\eta_j < 0$, so all these neurons are in resting state.

With $I(t) = 0$, most of the neurons are at rest. When $I(t)$ increases to 9, we can interpret it as if η_j increases by 9 units, so all of those neurons that were near the bifurcation point, start oscillating (672 out of 698 neurons in this example). The neurons in that region have very similar properties, as the heterogeneity is most noticeable at the tails of the quantile. This means that their initial values (their resting points, unless they have been perturbed) when $t_0 = 20$ ms, are quite close, and will reach V_{peak} at similar times, as their periods will be close as well. However, this sudden increase of activity and high amount of perturbations

in a short period does not seem to elicit many immediate spikes (black dots in fig.3.5). For reference, let us focus on one of these neurons that do spike due to an impulse: neuron $j = 660$, which is marked with a black dot before $t = 40$ ms. Its period is $T_{660} \simeq 10.341$ ms, and almost 1 ms before it spikes (0.997 ms), its phase is $\theta_{660} \simeq 9.1407$, so it is already close to spike. During this millisecond, there is an almost vertical line in the raster plot right before the spike, which translates to a chain of many small perturbations, but we see that they have little effect on the advance of the phase (they advance it approximately 0.203 ms). So this neuron finally fires not because the pulse it receives increases considerably its phase, but because it was already very close, and any small perturbation would have been enough. We will expand on this in the following section, where we may be able to better understand what happens right before and after a spike.

3.2.2.3 Study of synchronization through the PRCs

We have just seen that after adding a high enough external step current, there is some partial synchronization, but it cannot be maintained, as some neurons have faster dynamics than others. However, as we have advanced before, knowing the shape of the phase response curve can help us achieve synchronous firing from the population of neurons. Up until now, we have considered instantaneous synapses, as seen in [16], which means that when there is a spike, the signal is transmitted at the same moment instead of taking some time to reach the other neurons. According to [5], synaptic time delays can favor the emergence of synchronized states in the network, hence, we will adapt alg.1 to consider synaptic delays. It can be modeled by adding an extra equation that governs $s(t)$ (see [5]), but simply imposing a fixed time delay D_{syn} (i.e. we take $s(t - D_{syn})$) works as well in this case, and it can be implemented in the algorithm. To do so, once a neuron spikes, instead of perturbing the rest of the neurons, we will register it and assign it a timer starting at D_{syn}. Then, at the start of each iteration, we will check whether a neuron fires or the timer reaches zero faster. In the latter case, we perturb the neurons and, if as a result of that, a neuron spikes, we register it and start its corresponding timer, just as before. This way, we do not need the inner *while* loop we previously had, as we do not have infinitely fast synapses. Of course, we still need to be careful with the advances of time around the points where the external current is introduced or removed. Implementing that, we obtain the raster plot in fig.3.7.

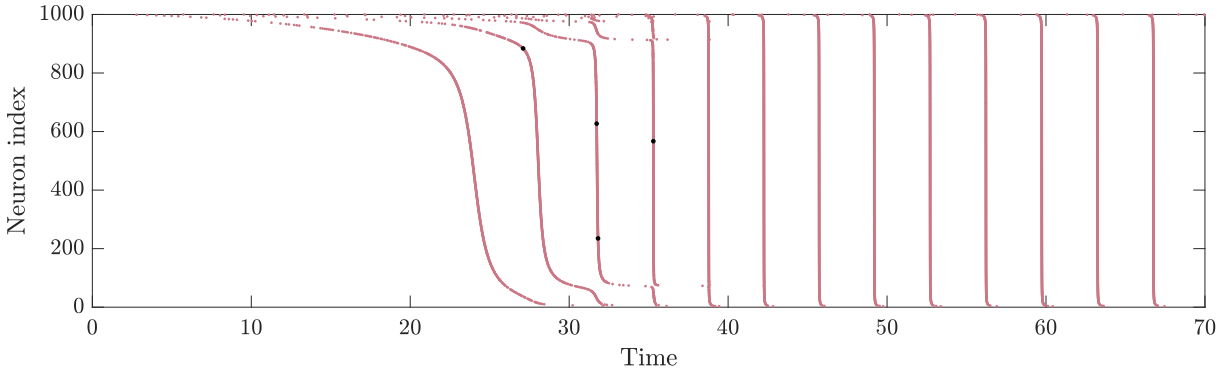

Figure 3.7: Raster plot of the same network as the one used in fig.3.5 but adding a synaptic delay ($D_{syn} = 3$ ms).

We note that the last neurons fire slightly faster that the synchronized group. Looking at fig.3.6, we can clearly see that their intrinsic current is considerably higher than that of the rest of the cells, so their period is shorter than the one for the synchronized line, and we have already seen that the perturbations cannot decrease the phase. A similar argument can be made for the first neurons, as they are too slow and not even the perturbations can advance enough their phase.

An interesting thing we have observed during the simulations is that, depending on the parameters used (particularly for for negative $\overline{\eta}$ and high synaptic strength J), the population of neurons is bistable. Initially, when $I(t) = 0$, the network exhibits low activity, as most of the neurons are at rest. With the addition of a strong external current, the vast majority of them end up synchronizing, and even when we remove this input, the strong synaptic coupling maintains the synchrony. So, depending on the initial conditions, we have the low activity state or the oscillatory state, as seen, for example, in fig.3.8. The network in fig.3.7 actually has only one steady state, as it does not really need an external input to force the resting neurons to spike. In this case, most of the them being already close to the bifurcation point, together with a strong synaptic strength, suffices.

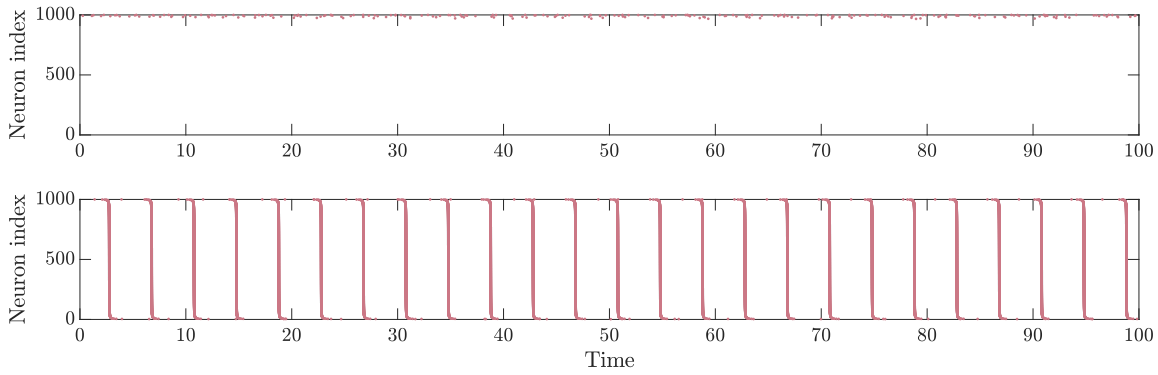

Figure 3.8: (Top) Network exhibiting low activity when all the neurons are initialized at V_{reset} and (bottom) the same network showing sustained oscillations when the neurons are initialized already synchronized. Parameters: $\overline{\eta} = -3$, $J = 13$, $\Delta = 0.3$, $D_{syn} = 3$.

With this bistability, one might think that we are in the bistable region in the bifurcation diagram presented in fig.3.4. However, by adding the delay, that diagram is not valid anymore, and furthermore, in that context, the bistability consisted in the coexistence of two stable fixed points[16] and not a stable limit cycle and a stable fixed point. For more details on the stability of the FREs with synaptic delays, we refer to [5].

Now, going back to the network simulated in fig.3.7, with alg.1 we can select a few neurons distant enough in terms of η_j, and see how their phase advances if we focus on the interval delimited by two of the vertical lines in fig.3.7, corresponding to the synchronized spikes. Storing the phase of the neuron every time it receives an impulse, we obtain fig.3.9. As we noted before, both the phase and the PRC have been normalized to lie between 0 and 2π, which is useful to compare between the different neurons, since they now have the same

domain and range. The vertical lines mark the phase at which the neuron receives a stimulus. These perturbations happen at the same time for each neuron, but not at the same phase.

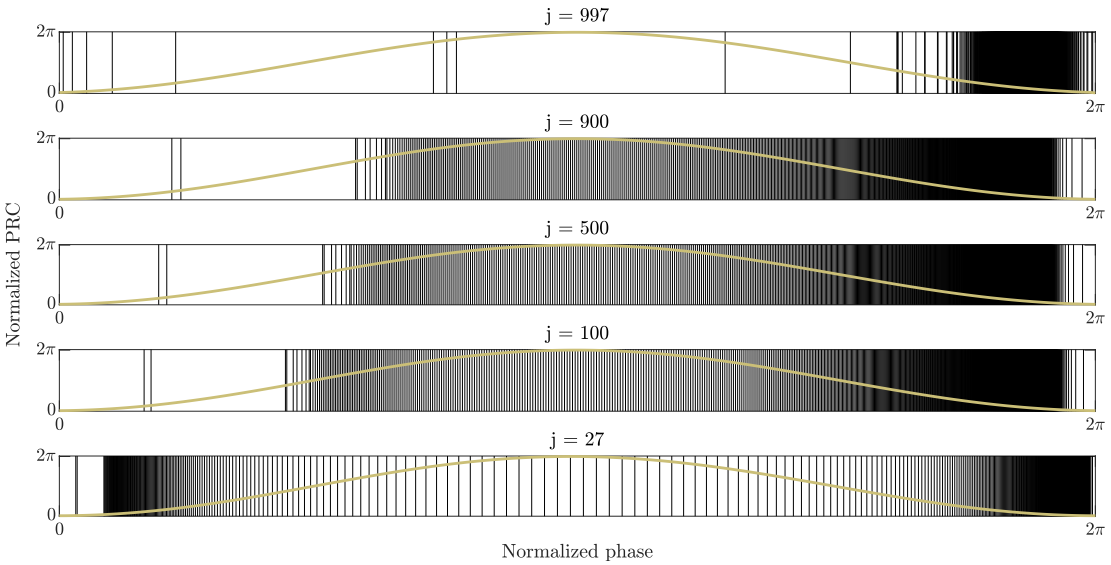

Figure 3.9: Normalized PRC (in yellow) of some selected neurons. The vertical lines mark the phase at which the neuron receives a stimulus. The inputs received before the denser part of perturbations correspond to spikes of the last neurons (the ones with higher index), as they spike several times before the synchronized neurons fire again. In particular, in the first plot, as the neuron spikes slightly earlier, we can even register the spikes made by the first neurons (the ones with lower index), which fire a bit later than the synchronized collective.

We observe that the chain of perturbations start at a later (normalized) phase for the neurons with faster dynamics, i.e. the ones with larger intrinsic current (see top panels in fig.3.9). Therefore, the neurons with higher frequency will be perturbed right at the end of their phase, where its impact will be minimal and, with lower frequencies, the pulses will land closer to the peak of the PRC, where they will be more effective. This way, the faster neurons will not deviate as much from their free-running trajectory, and the slower will be advanced to catch up with them. Nonetheless, we notice that for the cells around $j = 900$, which also have a high firing rate, the perturbations fall mostly in the interval where they are most effective, so their phase should advance even faster. However, the phase increase due to a single pulse in that region is insignificant anyway, since these neurons have a short slow region due to their fast dynamics which, in turn, means that their PRC will not reach large values. This is translated in fig.3.9 as the vertical lines being really close and, as we move to the less active neurons, the distance between them is more perceptible. This brings us to the next observation, which is that similarly to what happens for the fast neurons, where the stimuli is received near the end of the phase ($\theta = 2\pi$), for the slower neurons the stimuli starts near $\theta = 0$, where is not as effective either. However, for these neurons with lower intrinsic current, a single pulse can be very striking, even at lower phases and, if these initial pulses can push the phase close to where the PRC has its maximum, every perturbation will advance it drastically, as we can see in the last plot, where this can be interpreted as the vertical lines having a quite noticeable separation between them. For the

first neurons ($j < 27$) we cannot strictly talk about their PRCs, as they are not oscillating, but their dynamics are so slow, that the perturbations arrive too early to effectively advance their trajectory.

All of this explains the behaviour of the network in the previous section, when the synaptic delay was infinitely fast. When we had the almost vertical line around $t = 20$ ms, those neurons received the stimuli instantaneously, therefore, the perturbation was right at the beginning or the end of their phase, as they had just spiked or were about to do so. As we have seen, at those points the effect is almost non-existent, thus the neurons cannot synchronize. This also explains the black dots we discussed earlier, as the perturbations will make the neuron spike because it was already close to doing so, not because the phase was greatly advance, just as we observed. The same argument can be done for really small delays, which will not synchronize the neurons either. Moreover, if we increase the delay, the frequency may change, until the delay is so large that the synchrony disappears. This is known as *frequency suppression*[5].

4 Conclusions

In this project, we have studied how neurons interact with each other in a network. To do so, we have worked with the quadratic integrate-and-fire model for a single neuron, to which we have analyzed its dynamics and computed its analytical solution depending on the applied current. We have seen that, when the neuron exhibits periodic spiking, we can compute its phase response curve. In our case, as we had the exact solution of the QIF ODE, we were able to obtain the exact expression for its PRC which, for most models, is usually not possible.

Then, after reviewing a model for a network of neurons and a recently derived set of firing-rate equations (both for QIF neurons), we have designed an algorithm to describe a population of QIF neurons through their PRCs. Comparing with the aforementioned models, we have seen that our method accurately describes the dynamics of the network. With it, we were able to study how all the stimuli received from the network affects each neuron. In particular, we wanted to see whether a spike (or a quick succession of them) could advance the phase of a neuron enough to force it to spike. We reached the conclusion that, in those instances, the neuron did not spike due to a large advance of its phase, but because it was already really close to do so, and any small perturbation would have sufficed. Moreover, any pulse received near the end of the phase, will not have any significant effect, as the PRC is almost zero. This further explained our observations.

After that, we were interested in the acquisition of a population exhibiting synchronized firing. In chapter 2, when we studied the PRC of a QIF neuron, we observed that this model has an always positive PRC for perturbations greater than zero, which means that the phase will always be advanced for these kind of pulses. Furthermore, we noticed that the quadratic integrate-and-fire, when oscillating, has a slow region around half its period, where its membrane potential increases at a much lower pace. Here is where the PRC is greater, and so, it is where the phase makes longer jumps for any perturbation. Then, with our algorithm, we have seen how, if we delay the perturbations instead of having instant synapses, they have a more pronounced effect on the neurons with slower frequency, as they fall around the peak of the PRC. In addition to that, the PRC for the slower neurons has much higher values, so their phases advance much faster to catch up with the rest. All these factors combined, explain how the population can maintain the synchronized firing. So, with the tools we have studied and developed throughout the project, we managed to gain more insight into the generation of these coordinated states in a collective of neurons.

4.1 Future work

While we were able to use the exact solution of the QIF equation, this does not restrict the use of the tools developed in this project. The results obtained here validate the use of the PRCs and create a solid base to study the same problem with networks of neurons with expressions that we cannot explicitly integrate, which is usually the case. In this regard, we provide a proof of concept for the systematic use of PRCs in networks to tackle synchronization problems.

Due to the time constraints of a bachelor's degree thesis, it was not possible, but it would have been interesting to work in the more analytical part of the problem: finding a mean-field expression for the PRCs, following the same methodology as the authors in [16] used to derive the FREs for a network of QIFs (the *Ott-Antonsen* theory). The numerical simulations carried out in this project are a positive test that this might be feasible. Alongside, we could compute the PRC for the FREs of the QIF population, which, for the case of a limit cycle, can be done numerically with the adjoint method that we mentioned in section 2.1.1. For the case of the stable focus, we could use the theory developed in [17]. The final objective would be to compare the PRC of the FREs with the "mean-field PRC".

4.2 Acquired knowledge

Prior to this work, I had already studied single-neuron models, namely the Hodgkin-Huxley and FitzHugh-Nagumo equations and, more recently, I have also worked with the Morris-Lecar model. I have found these models very interesting and useful when studying the behaviour of excitable cells, but they have also doubled as great examples to introduce, or work more in-depth, important concepts in dynamical systems theory, such as bifurcations, stability and phase space analysis. In this project, working with the quadratic integrate-and-fire model has helped me further understand these concepts, and also see the importance of having simple models and why they are sometimes preferred over the more biophysically accurate ones. Here, I have also been introduced to the modeling of neuronal networks, with both the microscopic and the mean-field approaches. Apart from the theoretical results, seeing through the phase and the PRCs how the neurons communicate with each other and everything else that is written in this document, working with very high dimensional systems of thousands of neurons, has taught me to be more careful when writing code, particularly when creating arrays, since the memory quickly running out and the simulations taking too long to finish, were common issues I had to deal with. In summary, with this project, I have learned about many aspects related to mathematical modeling, and it has led me to discover many more interesting topics related to the vast field of mathematical neuroscience.

References

[1] Borisyuk A., Friedman A., Ermentrout B., Terman D. (2005) Tutorials in Mathematical Biosciences I: Mathematical Neuroscience. Berlin Heidelberg: Springer-Verlag

[2] Brown E., Moehlis J., Holmes P. (2004) On the phase reduction and response dynamics of neural oscillator populations. Neural Comput. 16:673-715.

[3] Buck J., Buck E., Case J.F., Hanson F.E. (1981) Control of flashing in fireflies. V. Pacemaker synchronization in Pteroptyx cribellata. J Comp Physiol. 144:287-298

[4] Castejón O. (2010) Analytical and computational tools for the study of phase and synchrony in neural oscillators. Master diss. Universitat Politècnica de Catalunya, Barcelona.

[5] Devalle F. (2019) Collective phenomena in networks of spiking neurons with synaptic delays. PhD diss. Universitat Pompeu Fabra, Barcelona.

[6] Ermentrout G.B., Kopell N. (1986) Parabolic bursting in an excitable system coupled with a slow oscillation. SIAM-J.-Appl.-Math, 46:233-253.

[7] Ermentrout B., Terman D. (2010) Mathematical Foundations of Neuroscience. New York: Springer-Verlag

[8] FitzHugh R. (1961) Impulses and physiological states in theoretical models of nerve membrane. Biophysical J. 1:445-466

[9] Hodgkin A.L., Huxley A.F. (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117:500-544

[10] Hoppensteadt F. C., Izhikevich E. M. (1997) Weakly Connected Neural Networks. New York: Springer-Verlag.

[11] Izhikevich E. M. (2000) Neural excitability, spiking, and bursting. International Journal of Bifurcation and Chaos, 10:1171-1266.

[12] Izhikevich E. M. (2003) Simple Model of Spiking Neurons. IEEE Transactions on Neural Networks, 14:1569-1572.

[13] Izhikevich E.M. (2007) Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. The MIT Press. (Chapter 10 can be found in the author's webpage https://www.izhikevich.org/publications/dsn/index.htm [Last Accessed: 30-11-2019])

[14] Lapicque L. (1907) Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. Journal de Physiologie et de Pathologie Générale 9:620-635

[15] Latham P., Richmond B., Nelson P., Nirenberg S. (2000) Intrinsic dynamics in neuronal networks. I. Theory. Journal of Neurophysiology, 83(2):808–827

[16] Montbrió E., Pazó D., Roxin A. (2015) Macroscopic Description for Networks of Spiking Neurons. Phys. Rev. X, 5:021028.

[17] Moreno R. (2019) Phase response to stimuli: the case of excitable systems. Master diss. Universitat Politècnica de Catalunya, Barcelona.

[18] Nagumo J., Arimoto S., Yoshizawa S. (1962) An active pulse transmission line simulating nerve axon. Proc. IRE. 50:2061-2070

[19] Shelley M. J., Tao L. (2001) Efficient and accurate time-stepping schemes for integrate-and-fire neuronal networks. Journal of Computational Neuroscience, 11.2:111–119.

A Nondimensionalization of the quadratic integrate-and-fire model

In 2000, Latham et al.[15], proposed a more general form of the quadratic integrate-and-fire, which can be written as

$$C_m\dot{V} = I + \overline{g}_l\frac{(V - V_r)(V - V_{th})}{V_{th} - V_r}, \tag{A.1}$$

where C_m and $\overline{g}_l$ are the membrane capacitance and the leak conductance respectively (same as with the Hodgkin-Huxley model), and V_r, V_{th} are the rest and threshold potentials.

Using the change of variables $V = \tilde{V}(V_{th} - V_r) + \frac{V_{th}+V_r}{2}$, (A.1) transforms into

$$\tau_m\dot{\tilde{V}} = \frac{I}{\overline{g}_l(V_{th} - V_r)} - \frac{1}{4} + \tilde{V}^2,$$

where $\tau_m = C_m/\overline{g}_l$ is the membrane time constant ($[C_m] = M^{-1}L^{-2}T^4I^2$, $[\overline{g}_l] = M^{-1}L^{-2}T^3I^2$ $\implies$ $[\tau_m] = T$). So, taking $\tilde{I} = \frac{I}{\overline{g}_l(V_{th}-V_r)} - \frac{1}{4}$, we end up with

$$\tau_m\dot{\tilde{V}} = \tilde{I} + \tilde{V}^2. \tag{A.2}$$

We note that $\tilde{V}$ has no dimensions. Conductance can be interpreted as the inverse of the resistance, so $\tilde{I}$ has no dimensions either. At the same time, we could have applied the change $t = \tilde{t}/\tau_m$, but we wanted to show equation (A.2), because considering the membrane time constant can sometimes be useful, as we will see throughout the document. Using this last change, we finally get

$$\dot{\tilde{V}} = \tilde{I} + \tilde{V}^2, \tag{A.3}$$

which is the nondimensionalized form of the QIF (in the main body of the document, we drop the tildes for notation's sake). This expression is actually the normal form of a saddle-node bifurcation (considering that the constant multiplying the term $\tilde{V}^2$ is 1). In fact, Hoppensteadt and Izhikevich showed in [10] that any neuronal model close to a saddle-node bifurcation can be transformed into (A.3) by a piecewise continuous change of variables.

Finally, if the reader is interested, we use the rest of this section to show the full derivation of the solutions for (A.3), presented in section 1.2.1. It is a Riccati's equation (in reduced

form), so it can be solved using the change $z = \frac{1}{V-\sqrt{-I}}$, but it is straightforward solving by separation of variables: For $I > 0$

$$\frac{dV}{dt} = I + V^2 \iff \int \frac{dV}{I+V^2} = \int dt.$$

The integral on the left-hand side can be expressed as

$$\frac{1}{\sqrt{I}} \int \frac{1/\sqrt{I}}{1+(V/\sqrt{I})^2}\, dV$$

thus

$$\frac{1}{\sqrt{I}} \arctan\left(\frac{V}{\sqrt{I}}\right) = t + C$$

and solving for V, we get

$$V(t) = \sqrt{I} \tan\left(\sqrt{I}(t+C)\right).$$

If we impose the initial condition $V(0) = V_0$, we get

$$V_0 = \sqrt{I} \tan\left(C\sqrt{I}\right) \iff C = \frac{1}{\sqrt{I}} \arctan\left(\frac{V_0}{\sqrt{I}}\right)$$

and the solution of the IVP is

$$V(t) = \sqrt{I} \tan\left(\arctan\left(\frac{V_0}{\sqrt{I}}\right) + t\sqrt{I}\right).$$

Now, for $I < 0$, we first perform the change of variables $I = -\tilde{I}^2$ ($I = -\tilde{I}$ also works, but this way we do not have to carry the square root throughout the calculations). Then

$$\frac{dV}{dt} = -\tilde{I}^2 + V^2 \iff \int \frac{dV}{V^2 - \tilde{I}^2} = \int dt.$$

The left-hand side can be written as

$$\int \frac{dV}{(V+\tilde{I})(V-\tilde{I})} = \frac{1}{2\tilde{I}} \int \frac{dV}{V-\tilde{I}} - \frac{1}{2\tilde{I}} \int \frac{dV}{V+\tilde{I}}.$$

Therefore

$$t + C = \frac{1}{2\tilde{I}} \log\left(V - \tilde{I}\right) - \frac{1}{2\tilde{I}} \log\left(V + \tilde{I}\right) = \frac{1}{2\tilde{I}} \log\left(1 - \frac{2\tilde{I}}{V+\tilde{I}}\right)$$

and undoing the change and solving for V, we get

$$V(t) = \frac{2\sqrt{-I}}{1 - e^{2\sqrt{-I}(t+C)}} - \sqrt{-I}.$$

Once again, if we impose the initial condition $V(0) = V_0$, we have

$$V_0 = \frac{2\sqrt{-I}}{1 - e^{2C\sqrt{-I}}} - \sqrt{-I} \iff C = \frac{1}{2\sqrt{-I}} \log\left(1 - \frac{2\sqrt{-I}}{V_0 + \sqrt{-I}}\right).$$

So the solution for the IVP with $I < 0$ is

$$V(t) = \frac{2\sqrt{-I}}{1 - e^{2t\sqrt{-I}}\left(1 - \frac{2\sqrt{-I}}{V_0 + \sqrt{-I}}\right)} - \sqrt{-I}.$$

Finally, for $I = 0$

$$\frac{dv}{dt} = V^2 \iff \int \frac{dV}{V^2} = \int dt \iff \frac{1}{V} = C - t \iff V(t) = \frac{1}{C - t}$$

and similarly to the previous cases, the solution of the initial value problem is

$$V(t) = \frac{V_0}{1 - V_0 t}.$$

And if we undo the initial changes of variables, we obtain the solutions for (A.1) depending on the sign of I.

B Deduction of the parametric curves in fig.3.4

Consider the system of firing-rate equations with the addition of the membrane time constant τ_m[5]

$$\tau_m \dot{r} = \frac{\Delta}{\pi \tau_m} + 2rv, \tag{B.1a}$$

$$\tau_m \dot{v} = v^2 + \overline{\eta} - (\pi \tau_m r)^2 + J\tau_m r. \tag{B.1b}$$

Notice that we consider $I(t) = 0$ (we can interpret $I(t)$ as an increase or decrease of $\overline{\eta}$). Furthermore, if $\tau_m = 1$, we recover the system (1.5), but we will see that this parameter does not affect the final result.

The system has 4 parameters, so we propose a change of variables to reduce this number:

$$v = \sqrt{\Delta}\tilde{v}, \qquad r = \frac{\sqrt{\Delta}}{\tau_m}\tilde{r}, \qquad t = \frac{\tau_m}{\sqrt{\Delta}}\tilde{t}.$$

With this, (B.1) is transformed into

$$\dot{\tilde{r}} = \frac{1}{\pi} + 2\tilde{r}\tilde{v}, \tag{B.2a}$$

$$\dot{\tilde{v}} = \tilde{v}^2 + \frac{\overline{\eta}}{\Delta} - (\pi\tilde{r})^2 + \frac{J}{\sqrt{\Delta}}\tilde{r}. \tag{B.2b}$$

So now we can study the stability considering the two parameters $\frac{\overline{\eta}}{\Delta}$ and $\frac{J}{\sqrt{\Delta}}$. The fixed points satisfy equations (B.2a)−(B.2b) when they are equated to 0 (i.e. they are the intersecting point of the nullclines), therefore, they verify

$$\frac{1}{\pi} + 2\tilde{r}\tilde{v} = 0, \tag{B.3}$$

$$\tilde{v}^2 + \frac{\overline{\eta}}{\Delta} - (\pi\tilde{r})^2 + \frac{J}{\sqrt{\Delta}}\tilde{r} = 0. \tag{B.4}$$

To study the linear stability of the fixed points, we need to compute the eigenvalues of the Jacobian matrix of the system evaluated at them. The matrix is given by

$$J = \begin{pmatrix} 2\tilde{v} & 2\tilde{r} \\ -2\pi^2\tilde{r} + \frac{J}{\sqrt{\Delta}} & 2\tilde{v} \end{pmatrix}$$

and it has eigenvalues

$$\lambda_{\pm} = 2\tilde{v} + \pm\sqrt{-4\pi^2\tilde{r}^2 + 2\frac{J}{\sqrt{\Delta}}\tilde{r}}.$$

From [16], we know that the system undergoes a saddle-node bifurcation (see fig.B.1). Therefore, we need to check when one of the eigenvalues is 0. We note that the (rescaled) firing-rate variable $\tilde{r}$ has to be nonnegative for it to have physical meaning. As a consequence, from eq.(B.3) we deduce that $\tilde{v} < 0$. This implies that only eigenvalue λ_+ can be zero, so we get the equation[1]

$$2\tilde{v} + \sqrt{-4\pi^2\tilde{r}^2 + 2\frac{J}{\sqrt{\Delta}}\tilde{r}} = 0. \tag{B.5}$$

So (B.3)−(B.4) together with (B.5) determine the points where the system undergoes a saddle-node bifurcation, and solving for $\left(\frac{\overline{\eta}}{\Delta}, \frac{J}{\sqrt{\Delta}}\right)$, we obtain the parametric curve

$$\left(\frac{\overline{\eta}}{\Delta}, \frac{J}{\sqrt{\Delta}}\right) = \left(-(\pi\tilde{r})^2 - \frac{3}{(2\pi\tilde{r})^2}, 2\pi^2\tilde{r} + \frac{1}{2\pi^2\tilde{r}^3}\right).$$

For the curve that serves as a delimiter for the regions of stable nodes and stable foci, we need to check when the imaginary part of the eigenvalues disappears (for a 2D system, a focus has two complex conjugate values, while a node has two real eigenvalues with the same sign). Therefore, the fixed point has to verify

$$-4\pi^2\tilde{r}^2 + 2\frac{J}{\sqrt{\Delta}}\tilde{r} = 0$$

and together with (B.3) and (B.4), we obtain the curve

$$\left(\frac{\overline{\eta}}{\Delta}, \frac{J}{\sqrt{\Delta}}\right) = \left(-(\pi\tilde{r})^2 - \frac{1}{(2\pi\tilde{r})^2}, 2\pi^2\tilde{r}\right).$$

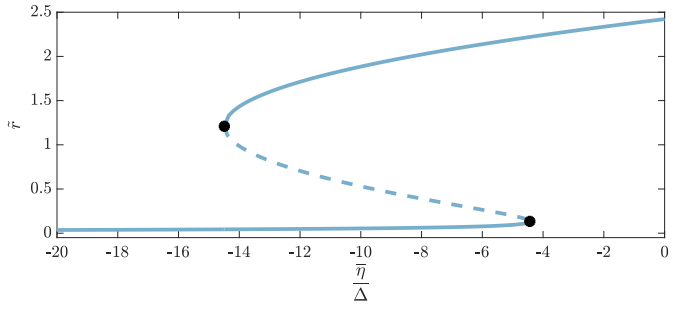

Figure B.1: Bifurcation diagram for $J = 20$ and $\Delta = 0.7$. The black asterisks mark the bifurcation points, and the dashed line corresponds to the unstable (saddle) fixed point.

[1] If we had not made this observation, to find if a fixed point is the locus of a saddle-node bifurcation, we can check when the determinant of the Jacobian evaluated at that point is zero. The determinant of a matrix is the product of its eigenvalues, so if it is 0, then one of them has to be 0.

C Matlab scripts

C.1 Network of QIF neurons

```
clear all; close all; clc;
tic;

%%% Parameters %%%

t0 = 0; tn = 80;

N = 1000;     %Number of neurons in the network
j = 1:N;      %Neuron index

etaB = -5;    %Bar eta
J = 15;
d = 0.8;      %Delta
eta = etaB + d*tan((pi/2)*((2*j - N - 1)/(N + 1)));
Vpeak = 100; Vreset = -100;
V = Vreset + (Vpeak-Vreset)*rand(1,N); %The initial voltage for each neuron

%%% Integration %%%

dt = 1e-4;
n = (tn-t0)/dt;
t1 = linspace(t0,tn,n);
I = I1(t1);
s = zeros(1,n);
refractory_time = [];                                   %Time steps left for neurons in ref time
refractory_indexes = [];                                %Indexes from V of neurons in ref time;
nSpikes = [0];
raster_indexes = 1:1000;
raster_spikes = zeros(n,1000);

for i = 0:n-1
    if i > 0
        V = V + dt*((V.^2) + eta + J*s(i) + I(i));      %Euler step
    end
    refractory_time = refractory_time - 1;              %Ref neurons are one step closer to freedom
    V(refractory_indexes) = Vreset;                     %The neurons in ref state cannot be updated
    Vpeak_indexes = find(V >= Vpeak);                   %Find indexes of the neurons that reached Vp
    raster_spike_indexes = find(V(raster_indexes) >= Vpeak); %The same again for the ones in the raster
    V(Vpeak_indexes) = Vreset;                          %Reset their voltage
    refractory_indexes = [refractory_indexes Vpeak_indexes]; %Add them into ref state
    refractory_time(end+1:end+size(Vpeak_indexes,2)) = 200; %Initialize their counter to 200 steps
    finish_ref_time = find(refractory_time == 0);       %Find the ones that just finished ref time
    refractory_indexes(finish_ref_time) = [];           %Remove them from the list
    refractory_time(finish_ref_time) = [];              %Remove them from the counter list as well
    spike_indexes = find(refractory_time == 100);       %Find the neurons that have reached +inf
    nSpikes(i+1) = size(spike_indexes,2);               %At +inf, we say that they have spiked
    raster_spikes(i+1,raster_spike_indexes) = 1;

    if i < 10
        s(i+1) = sum(nSpikes)/(N*1e-3);
    else
        s(i+1) = sum(nSpikes(end-9:end))/(N*1e-3);
    end
end
toc;

%%% Plots %%%

figure;
for i = 1:1000
    p = find(raster_spikes(:,i) == 1);
    if size(p,1) > 0
        %For the raster plot, the spikes are counted once Vpeak is reached, but the spike is actually
        %fired 10^-2 time units after that
        plot(t1(p)+1e-2, i, '.b', 'MarkerSize', 1); hold on;
    end
end
```

```
68
69  ylabel('Neuron index'); xlabel('Time');
70  xlim([t0,tn]);
71
72
73
74  %%% Functions %%%
75
76  function res = I1(t)
77      s = max(size(t));
78      res = zeros(1,s);
79      aux = find(t >= 20 & t <= 50);
80      res(aux) = 3;
81  end
```

C.2 Algorithm for coupling PRCs

```
1   clear all; close all; clc;
2   tic;
3
4   %%% Parameters %%%
5
6   t0 = 0; tn = 80; %Initial and final time
7
8   N = 1000;          %Number of neurons in the network
9   j = (1:N)';        %Neuron index
10
11  etaB = -5;         %Bar eta
12  J    = 15;
13  d    = 0.8;        %Delta
14  eta  = etaB + d*tan((pi/2)*((2*j - N - 1)/(N + 1)));
15  Vp   = 100; Vr = -100; %Vpeak and Vreset
16
17  te_0 = 20; te_n = 50; %Start and end time for the external input I(t)
18
19  %%% Algorithm %%%
20
21  spNeurons1 = find(eta + 0 > 0);          %Indexes of the neurons that exhibit periodic spiking when I(t) =
        0, i.e. the neurons with Ij = eta + 0 > 0.
22  spNeurons2 = find(eta + I(te_0) > 0);  %The same but when I(t) = c > 0
23  spNeurons  = spNeurons1;               %This vector will change to spNeurons2 when I(t) is activated, and
        will return to spNeurons1 when deactivates
24
25  chNeurons = find((eta + 0 < 0) & (eta + I(te_0) > 0)); %These are the indexes of the neurons that at first
        would be resting, but the increase of I(t) makes them produce periodic spikes
26
27  restNeurons1 = find(eta + 0 < 0);        %The same but with the neurons that are at rest
28  restNeurons2 = find(eta + I(te_0) < 0);
29  restNeurons  = restNeurons1;
30
31  Vrest1 = -sqrt(-eta(restNeurons));       %The stable fixed point for the restNeurons.
32  Vrest2 = -sqrt(-eta(restNeurons2)-I(te_0));
33  Vrest  = Vrest1;
34
35  %For the simulations carried on this project, there were no neurons with eta_j = 0. Otherwise, vectors
        similar to spNeurons and restNeurons should be included
36
37  theta = 0*(2*pi)*rand(size(spNeurons,1),1); %Random initial values for the phase of the spiking neurons.
        theta \in [0,2pi)
38
39  Tj = T(eta(spNeurons) + I(t0),Vr,Vp);      %Period for each neuron
40
41  V0 = -100 + 0*200*rand(size(restNeurons,1),1); %Random initial values for the voltage of restNeurons. We
        need to keep track of their voltage in case some perturbation forces them to produce a spike.
42
43  S = zeros(N,1);
44
45  i = 1;      %iteration counter
46  t = t0;     %Time
47
48  while t < tn
49      tk = min(Tj - (theta/(2*pi)).*Tj);
50      tk2 = min(T(eta(restNeurons) + I(t), V0, Vp));
51      aux = 0;
52
53      if (t < te_0) && (t + min(tk,tk2) >= te_0) %The external input I(t) "turns on"
54          theta = (theta/(2*pi)).*Tj;   %We "denormalize" theta using the old period Tj
55          theta = theta + (te_0 - t);   %All the phases increase until te_0 is reached, and from there, we
                will compute the time until the next spike, considering the new period.
56          V0 = V(te_0 - t,0,V0,eta(restNeurons) + I(t),Vp,Vr,Vrest); %The same for the membrane potential of
                the other neurons.
57          V0sp = sqrt(eta(spNeurons) + I(t)).*tan(atan(Vr./sqrt(eta(spNeurons) + I(t))) + sqrt(eta(spNeurons
                ) + I(t)).*theta); %We compute the membrane potential of the spiking neurons to recompute
                their phase in the next line.
```

```
58          theta = [T(eta(chNeurons) + I(t + tk),Vr,Vp) - T(eta(chNeurons) + I(t + tk),V0(end-size(chNeurons
                ,1)+1:end),Vp); T(eta(spNeurons) + I(t + tk),Vr,Vp) - T(eta(spNeurons) + I(t + tk),V0sp,Vp)];
                %With the increase of intensity, chNeurons now exhibit periodic spiking, so we need to compute
                 their phase. We also recompute the phase of the spiking neurons.
59
60          spNeurons = spNeurons2;
61          restNeurons = restNeurons2;
62          Vrest = Vrest2;
63
64          Tj = T(eta(spNeurons) + I(t + tk),Vr,Vp); %The period is recomputed
65          V0(end-size(chNeurons,1)+1:end) = [];      %At [te_0, te_n] we do not keep track of the membrane
                potential of chNeurons
66          tk = min(Tj - theta) + (te_0 - t);
67          tk2 = min(T(eta(restNeurons) + I(t + tk), V0, Vp)) + (te_0 - t);
68          aux = te_0 - t;
69
70          theta = (theta./Tj)*2*pi;          %We normalize theta with the new period Tj
71
72      elseif (t <= te_n) && (t + min(tk,tk2) > te_n) %The external input I(t) "turns off"
73          theta = (theta/(2*pi)).*Tj;
74          theta = theta + (te_n - t);
75          V0 = V(te_n - t,0,V0,eta(restNeurons) + I(t),Vp,Vr,Vrest);
76
77          spNeurons = spNeurons1;
78          restNeurons = restNeurons1;
79          Vrest = Vrest1;
80
81          Tj = T(eta(spNeurons) + I(t + tk),Vr,Vp);
82          V0 = [V0; sqrt(eta(chNeurons) + I(t)).*tan(atan(Vr./sqrt(eta(chNeurons) + I(t))) + sqrt(eta(
                chNeurons) + I(t)).*theta(1:size(chNeurons,1)))];
83          theta(1:size(chNeurons,1)) = [];
84          V0sp = sqrt(eta(spNeurons) + I(t)).*tan(atan(Vr./sqrt(eta(spNeurons) + I(t))) + sqrt(eta(spNeurons
                ) + I(t)).*theta);
85          theta = T(eta(spNeurons) + I(t + tk),Vr,Vp) - T(eta(spNeurons) + I(t + tk),V0sp,Vp);
86          tk = min(Tj - theta) + (te_n - t);
87          tk2 = min(T(eta(restNeurons) + I(t+tk), V0, Vp)) + (te_n - t);
88          aux = te_n - t;
89
90          theta = (theta./Tj)*2*pi;
91      end
92
93      if tk < tk2
94          k = find(Tj - (theta/(2*pi)).*Tj + aux == tk);
95          Lt_sp = k; Lt_rest = [];
96          k = spNeurons(k);
97      elseif tk > tk2
98          tk = tk2;
99          k = find(T(eta(restNeurons) + I(t+tk), V0, Vp) + aux == tk);
100         Lt_sp = []; Lt_rest = k;
101         k = restNeurons(k);
102     else
103         k2 = find(T(eta(restNeurons) + I(t+tk), V0, Vp) + aux == tk);
104         k = find(Tj - (theta/(2*pi)).*Tj + aux == tk);
105         Lt_sp = k; Lt_rest = k2;
106         k = spNeurons(k);
107         k2 = restNeurons(k2);
108         k = [k2; k];
109     end
110
111     S(k,i) = tk;
112
113     theta = theta + (tk./Tj)*2*pi + ((PRC((theta/(2*pi)).*Tj, (size(k,1)*J)/N, eta(spNeurons) + I(t + tk),
            Vr, Tj))./Tj)*2*pi;
114     theta(Lt_sp) = 0;
115     V0 = V(tk, 0, V0, eta(restNeurons) + I(t + tk), Vp, Vr, Vrest); V0 = V0 + (size(k,1)*J)/N;
116     V0(Lt_rest) = Vr;
117
118     L_sp = find(theta >= 2*pi); Lt_sp = [Lt_sp; L_sp]; theta(L_sp) = 0;
119     nSpikes = size(L_sp,1);
120     L_rest = find(V0 >= Vp); Lt_rest = [Lt_rest; L_rest]; V0(L_rest) = Vr;
121     nSpikes = nSpikes + size(L_rest,1);
122
123     while nSpikes ~= 0
124         S([restNeurons(L_rest); spNeurons(L_sp)], i) = tk + 1; %The +1 is to distinguish these neurons
                from the ones in k
125
126         theta = theta + ((PRC((theta/(2*pi)).*Tj, (nSpikes*J)/N, eta(spNeurons) + I(t + tk), Vr, Tj))./Tj)
                *2*pi;
127         theta(Lt_sp) = 0;
128         V0 = V0 + (nSpikes*J)/N;
129         V0(Lt_rest) = Vr;
130
131         L_sp = find(theta >= 2*pi); Lt_sp = [Lt_sp; L_sp]; theta(L_sp) = 0;
132         nSpikes = size(L_sp,1);
133         L_rest = find(V0 >= Vp); Lt_rest = [Lt_rest; L_rest]; V0(L_rest) = Vr;
134         nSpikes = nSpikes + size(L_rest,1);
135     end
136
137     i = i + 1;
138     t = t + tk;
139 end
140
```

```
141 toc;
142
143 %%% Plots %%%
144
145 figure;
146 t = t0;
147 for k = 1:i-1
148     v = find(S(:,k) ~= 0);  %I'm assuming that no neuron will fire exactly at t = 0...
149     tk = min(S(v,k));
150     first = find(S(:,k) == tk);
151     if tk == -1
152         tk = 0;
153     end
154     plot(t+tk, first,'.r', 'MarkerSize', 6);hold on;
155     t = t + tk;
156 end
157
158 %We separate the plots so that the ones that are marked in black fall on top of the others. This way they
      are more visible.
159
160 t = t0;
161 for k = 1:i-1
162     v = find(S(:,k) ~= 0);
163     tk = min(S(v,k));
164     if tk == -1
165         tk = 0;
166     end
167     v = find(S(:,k) == tk+1);
168     plot((t+tk)*(v./v), v, '.k', 'MarkerSize', 7);
169     t = t + tk;
170 end
171 xlim([t0 tn]);
172
173
174
175 %%% Functions %%%
176
177 function res = I(t)
178     res = 0;
179     if t >= 20 && t <= 50
180         res = 3;
181     end
182 end
183
184
185 function res = PRC(theta, A, I, Vr, T)
186     J = 1./sqrt(I);
187     res = min(J.*(atan(A*J + tan(atan(J*Vr) + theta.*sqrt(I))) - atan(J*Vr)), T) - theta;
188 end
189
190
191 function res = T(I,V0,Vp)
192     if I(1) > 0 %We suppose that all the neurons have Ij with the same sign
193         J = 1./sqrt(I);
194         res = J.*(atan(J*Vp) - atan(J.*V0));
195     elseif I(1) < 0
196         a = Vp*V0; b = Vp*sqrt(-I); c = V0.*sqrt(-I);
197
198         res = (1./(2*sqrt(-I))).*log((a + b - c + I)./(a - b + c + I)); %Time until next spike for the
              neurons with I < 0
199
200         aux = find(V0 <= sqrt(-I));
201         res(aux) = +inf;
202     else
203         res = (1./V0) - 1/Vp;
204         aux = find(V0 <= 0);
205         res(aux) = +inf;
206     end
207 end
208
209
210 function res = V(t,t0,V0,I,Vp,Vr, Vrest)
211     if I(1) < 0 %We suppose all the neurons have the same sign of the intensity, so we check only the
          first
212         res = (2*sqrt(-I))./(1-exp(2*sqrt(-I)*(t-t0)).*(1-((2*sqrt(-I))./(V0 + sqrt(-I))))) - sqrt(-I);
213     elseif I(1) == 0
214         res = V0./(1-V0*(t-t0));
215     else
216         res = sqrt(I).*tan(atan(V0./sqrt(I)) + sqrt(I)*(t-t0));
217     end
218
219     spIndex = find((V0 > -Vrest & res < Vrest) | (res >= Vp));
220     res(spIndex) = Vr;
221 end
```