Dynamical Systems in Neuroscience

Taravat Saeb Gilani and Philipp Hövel[1]
Institut für Theoretische Physik, Technische Universität Berlin
Hardenbergstraße 36, 10623 Berlin, Germany
Bernstein Center for Computational Neuroscience, Humboldt-Universität zu Berlin,
Philippstraße 13, 10115 Berlin, Germany
Date: July 2, 2013

Contents

[1]contact: phoevel@physik.tu-berlin.de

1. Preface

Based on preparations for several courses on nonlinear dynamics offered at *Technische Universität Berlin*, this text summarizes various mathematical models that have strong ties to neuroscience. The relevance of each of these models arises from their ability to address specific features that are characteristic to certain aspects of neural dynamics. For example, different responses to an external stimulus can be associated to different bifurcation scenarios. Thus, they can be adequately modeled by corresponding normal-form models.

With this documentation we aim to provide a list of generic models and their characteristic properties that are tailor-made for the individual demands of an experimental finding. The main focus is not a one-to-one correspondence to physiological quantities, but a phenomenological description that allows for application of tools well established and widely used in nonlinear dynamics. The goal is not a collection of physiological models that describe the neurological anatomy in detail, but an introduction to nonlinear systems related to neuroscience. Accordingly, the number of variables and parameters in the equations to be considered will be kept to a minimum while they still capture specific properties of neural behavior. Each chapter follows a similar structure, which facilitates a comparison of the different model equations and at the same time summarizes basic techniques on nonlinear dynamics.

At the end, this summary will enable the reader to select a model appropriate to his/her intentions to design a theoretical experiment or to reproduce results obtained by experiments. Exemplary MATLAB source code, which was used to generate some of the figures in this documentation, can be found in the appendix C.

This text is far from being completed. To the contrary, all chapters should be considered as work in progress and subject to change. Besides editing the existing material, we will also extend the text by additional chapters, for instance, on network science. We also appreciate feedback, comments, and remarks either via the website `http://www.itp.tu-berlin.de/?hoevel` (→ *Teaching* → *Documentation*) or by email `phoevel@physik.tu-berlin.de`.

2. SNIPER Model

In this chapter we consider a first generic model that is paradigmatic for a specific type of excitable dynamics. The behavior of the system is governed by a Saddle-node on a limit cycle (*Saddle-Node Infinite-PERiod bifurcation - SNIPER*, also called *SNIC: Saddle-Node on an Invariant Cycle*). Its dynamics is described by the following system of first-order differential equations

$$\dot{x} = x\left(1 - x^2 - y^2\right) + y\,(x - b) \tag{2.1a}$$
$$\dot{y} = y\left(1 - x^2 - y^2\right) - x\,(x - b)\,. \tag{2.1b}$$

x and y are the variables in appropriate center manifold coordinates and $b > 0$ denotes a bifurcation parameter. This parameter influences the type of dynamics and determines where in the (x, y)-plane the fixed points are located, as discussed below. These equations can be used for the description of an excitable element; in this case a neuron of excitability type I. We will discuss a model for type-II excitability later in chapter 3. Sometimes it is also insightful to rewrite Eqs. (2.1) in polar coordinates $(x = r\cos\varphi, y = r\sin\varphi)$:

$$\dot{r} = r\left(1 - r^2\right) \tag{2.2a}$$
$$\dot{\varphi} = b - r\cos\varphi. \tag{2.2b}$$

Excitability implies that the neuron activity is near a transition from quiescent state to periodic spiking. For type-I excitability the oscillations can have an arbitrary small frequency and finite amplitudes close, but above the bifurcation. Such a transition corresponds to saddle-node bifurcation on a limit cycle. In an experiment action potentials can be generated with arbitrary low frequency, depending on the strength of the applied current. Type-I neural excitability is observed when a rest potential disappears by means of a saddle-node bifurcation. Close to the bifurcation point, the frequency of the emerging orbit tends to zero and thus the spiking rate becomes arbitrarily low.

Before discussing exemplary time series arising from Eqs. (2.1), we will at first present details on the underlying bifurcation.

2.1. Bifurcation Analysis

In the following we will determine the fixed points of the SNIPER model given by Eqs. (2.1). Then we will subsequently discuss the stability of each fixed point.

Stationary solutions - fixed points: The fixed points (x_i^*, y_i^*) of the Eqs. (2.1) are the solutions of the following system of equations

$$x\left(1 - x^2 - y^2\right) + y\,(x - b) = 0 \tag{2.3a}$$
$$y\left(1 - x^2 - y^2\right) - x\,(x - b) = 0. \tag{2.3b}$$

First, there exists the trivial solution

$$(x_A^*, y_A^*) = (0, 0). \tag{2.4}$$

Using the information about this trivial fixed point, the other fixed points are given by the following equation

$$\left(1 - x^2 - y^2\right) = (x - b) = 0. \tag{2.5}$$

It is easy to find the two remaining fixed points.

$$(x_B^*, y_B^*) = \left(b, +\sqrt{1 - b^2}\right) \tag{2.6a}$$

$$(x_C^*, y_C^*) = \left(b, -\sqrt{1 - b^2}\right). \tag{2.6b}$$

The fixed points (x_2^*, y_2^*) and (x_3^*, y_3^*) exist only below and at the bifurcation, i.e., $b \leq 1$. They collide for $b = 1$ and vanish above the bifurcation.

In order to examine the stability properties of the fixed points, we perform a linear stability analysis of each of the three fixed points, that is, we determine the eigenvalues of the system's Jacobian matrix $\mathbf{J}$ at the coordinates the fixed points:

$$\mathbf{J} = \begin{pmatrix} 1 - 3x^2 + y - y^2 & x - 2xy - b \\ -2x - 2xy + b & 1 - x^2 - 3y^2 \end{pmatrix}. \tag{2.7}$$

This leads to the characteristic equation

$$\det\left(\mathbf{J} - \Lambda\, \mathbf{I_2}\right) = 0 \tag{2.8a}$$

$$\Leftrightarrow \quad \Lambda^2 - \Lambda\ \mathrm{tr}\mathbf{J} + \det \mathbf{J} = 0, \tag{2.8b}$$

where Λ denotes the eigenvalues, tr refers to the trace, and $\det$ is the determinant of the Jacobian matrix $\mathbf{J}$. $\mathbf{I_2}$ names the 2×2 identity matrix. Inserting the coordinates of the fixed points in $\mathbf{J}$ leads to the respective eigenvalues.

Unstable focus: By inserting the coordinates of the first fixed point $(x_A^*, y_A^*) = (0, 0)$ in Eq. (2.7) we obtain the Jacobian matrix

$$\mathbf{J}_f = \begin{pmatrix} 1 & -b \\ b & 1 \end{pmatrix}. \tag{2.9}$$

Solving the characteristic equation (2.8b)

$$0 = \Lambda^2 - 2\Lambda + b^2 + 1 \tag{2.10}$$

yields the eigenvalues of this fixed point

$$\Lambda_{1/2} = 1 \pm \sqrt{-b^2} \tag{2.11a}$$

$$= 1 \pm ib. \tag{2.11b}$$

The eigenvalues' real part is always greater than zero. Since there exists a complex conjugate pair of eigenvalues for $b \neq 0$, the fixed point is an unstable focus.

Saddle node: In analogy, we get for the second fixed point at $(x_B^*, y_B^*) = \left(b, \sqrt{1-b^2}\right)$

$$\mathbf{J}_s = \begin{pmatrix} -2b^2 + \sqrt{1-b^2} & -2b\sqrt{1-b^2} \\ -b - 2b\sqrt{1-b^2} & -2 + 2b^2 \end{pmatrix}. \tag{2.12}$$

Solving the characteristic equation

$$\begin{aligned} 0 &= \Lambda^2 - \Lambda\left(\sqrt{1-b^2} - 2\right) - 2\sqrt{1-b^2} && \text{(2.13a)} \\ &= (2+\Lambda)\left(\Lambda - \sqrt{1-b^2}\right) && \text{(2.13b)} \end{aligned}$$

we obtain the following eigenvalues

$$\Lambda_1 = -2, \qquad \Lambda_2 = \sqrt{1-b^2}. \tag{2.14}$$

For every $b \in [0,1]$ the eigenvalue Λ_2 is greater or equal zero and Λ_1 is always less than zero: $\Lambda_1 = -2 < 0 \leq \Lambda_2$. Therefore, the examined fixed point is a saddle node with one attracting and one repelling direction. For $b > 1$ the fixed point does not exist.

Stable node: Again, the stability analysis follows in analogy to the fixed points considered above. The Jacobian matrix reads for $(x_C^*, y_C^*) = \left(b, -\sqrt{1-b^2}\right)$

$$\mathbf{J}_n = \begin{pmatrix} -2b^2 - \sqrt{1-b^2} & 2b\sqrt{1-b^2} \\ -b + 2b\sqrt{1-b^2} & -2 + 2b^2 \end{pmatrix}. \tag{2.15}$$

The characteristic equation becomes

$$\begin{aligned} 0 &= \Lambda^2 - \Lambda\left(-\sqrt{1-b^2} - 2\right) + 2\sqrt{1-b^2} && \text{(2.16a)} \\ &= (2+\Lambda)\left(\Lambda + \sqrt{1-b^2}\right) && \text{(2.16b)} \end{aligned}$$

and its solutions are given by

$$\Lambda_1 = -2, \qquad \Lambda_2 = -\sqrt{1-b^2}. \tag{2.17}$$

For $b \in [0,1]$ the eigenvalue Λ_2 is less or equal zero and Λ_1 is always less than zero: $\Lambda_1 = -2 < \Lambda_2 \leq 0$. Therefore the fixed point $\left(b, -\sqrt{1-b^2}\right)$ is a stable node for all values $b \leq 1$. For $b > 1$ the fixed point does not exist.

To summarize the results of this linear stability analysis, we have determined the existence of an unstable focus at the origin. Furthermore, we found in the parameter $b \in (0,1)$ 2 additional fixed points, a saddle and a stable node. They collide at $b = 1$, annihilate, and give way for a limit cycle.

2.2. Numerical Simulations

Next we will provide and trajectories in phase space that illustrate the underlying bifurcation. We simulate the equations with explicit Euler method and our time step is 0.002. For details on this integration method see appendix B. The MATLAB code, which will be used in the following to generate the time series, can be found in appendix C.

Below the bifurcation, $b < 1$, $(b = 0.7)$**:**

Figure 2.1 displays the time series of the SNIPER model given by Eqs. (2.1) in the upper panels. Lower panel shows trajectories in the (x, y)-plane and the locations of the 3 fixed points.

Irrespective from the starting point, an arbitrary trajectory is attracted towards the unit circle and ends up in the stable node C. Starting above the stable manifold, that is, the attracting direction, of the saddle-point B induces a counter clockwise rotation as shown by the blue curve. The trajectory performs a complete excursion in the phase space before ending at the stable node. Starting below the stable manifold means a clockwise rotation directly into the fixed point C. See green curve in Fig. 2.1. For initial conditions inside the unit circle (red curve), the trajectory is repelled by the unstable focus A and finally approaches the stable node C.

At the bifurcation, $b = 1$**:** If the bifurcation parameter b increases, the saddle-point and the stable node approach each other. They collide at $(1, 0)$ for $b = 1$. At this point the saddle-point and the stable node annihilate each other, but the unstable focus at the origin remains (see Fig. 2.2). Thus, a limit cycle is about to form. In this case all trajectories rotate counter clockwise. However, no trajectory can perform more than one excursion in the phase space and all trajectories approach the fixed point at the end.

Above the bifurcation, $b > 1$, $(b = 1.05)$**:**

For $b > 1$ the saddle-point and the stable node have annihilated and the unstable node at the origin is the only fixed point left. A limit cycle on the unit circle form the sole attractor of the system. As opposed to the two cases above, now the system oscillates. From the polar-coordinate notation in Eqs.(2.2) it becomes clear that the unit circle $r = 1$ describes the limit cycle.

For further information read also Refs. [STR94a, HU93a, DIT94, HIZ07, HIZ07a, AUS09, AUS07, IZH07].

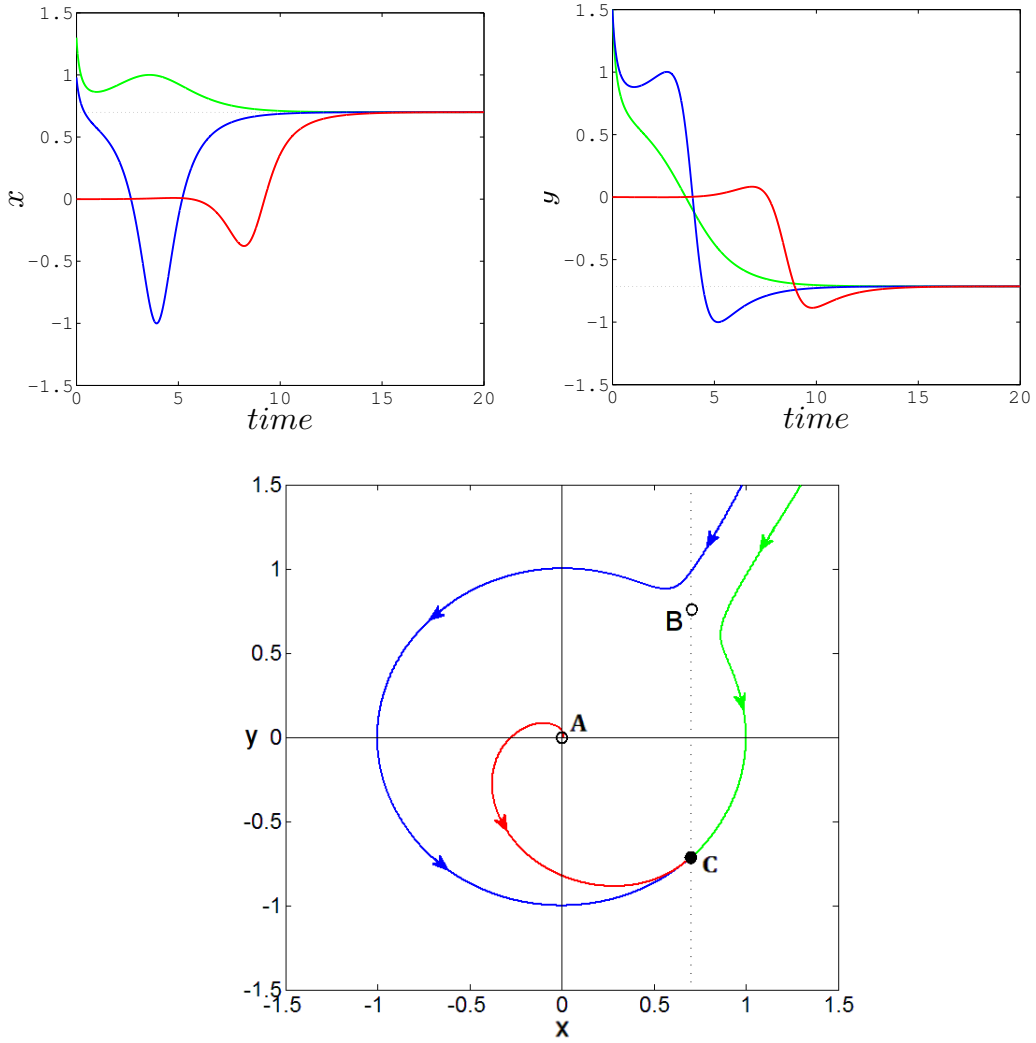

Fig. 2.1.: Time series and phase portrait below the bifurcation ($b < 1$): Points A, B, and C denote the unstable focus, the saddle point, and the stable node, respectively. The blue, green and red curves correspond to trajectories starting above, below the stable manifold of the saddle-point, and close to the focus, respectively.

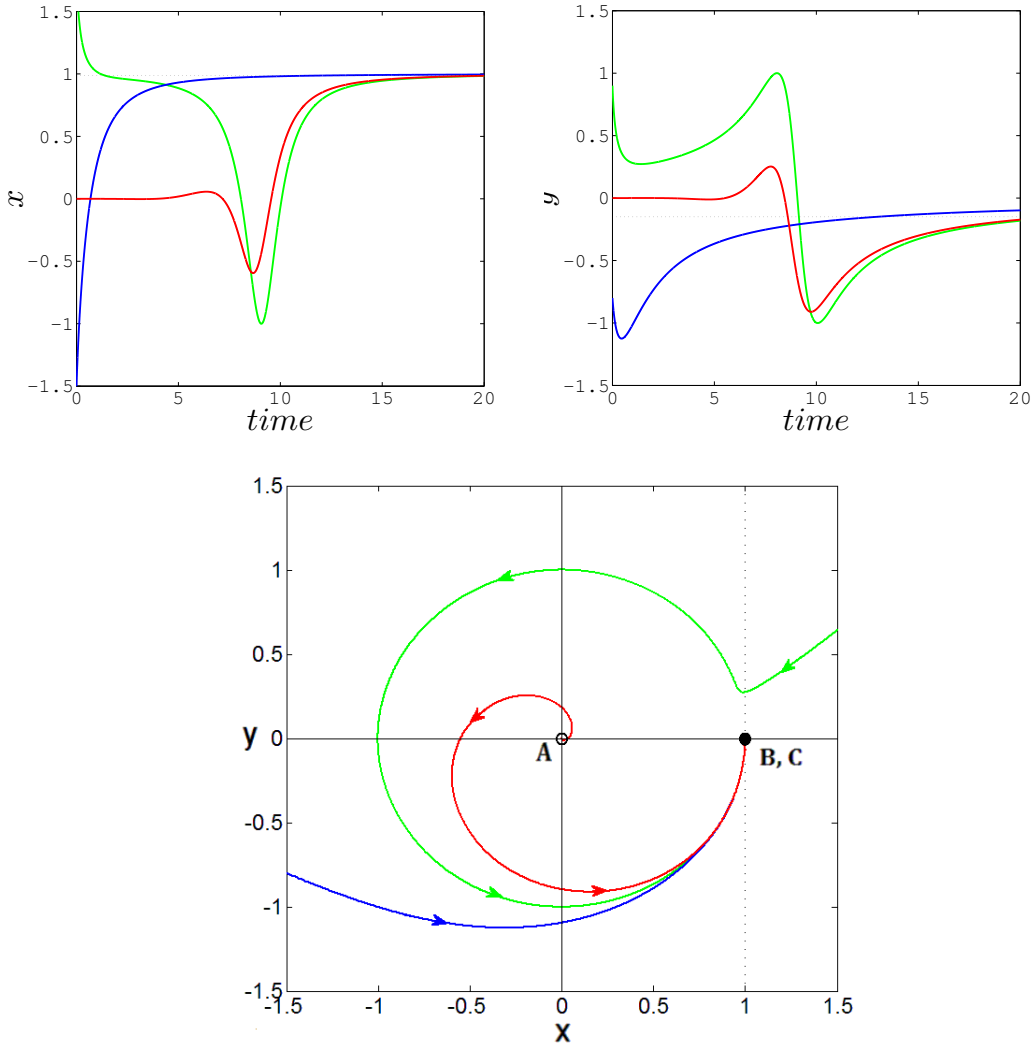

Fig. 2.2.: Time series and phase portrait at the bifurcation ($b = 1$): Point A denotes the unstable focus. Points B and C correspond to the collided saddle-point and stable node, respectively. Irrespective from their starting points, all trajectories rotate counter clockwise.

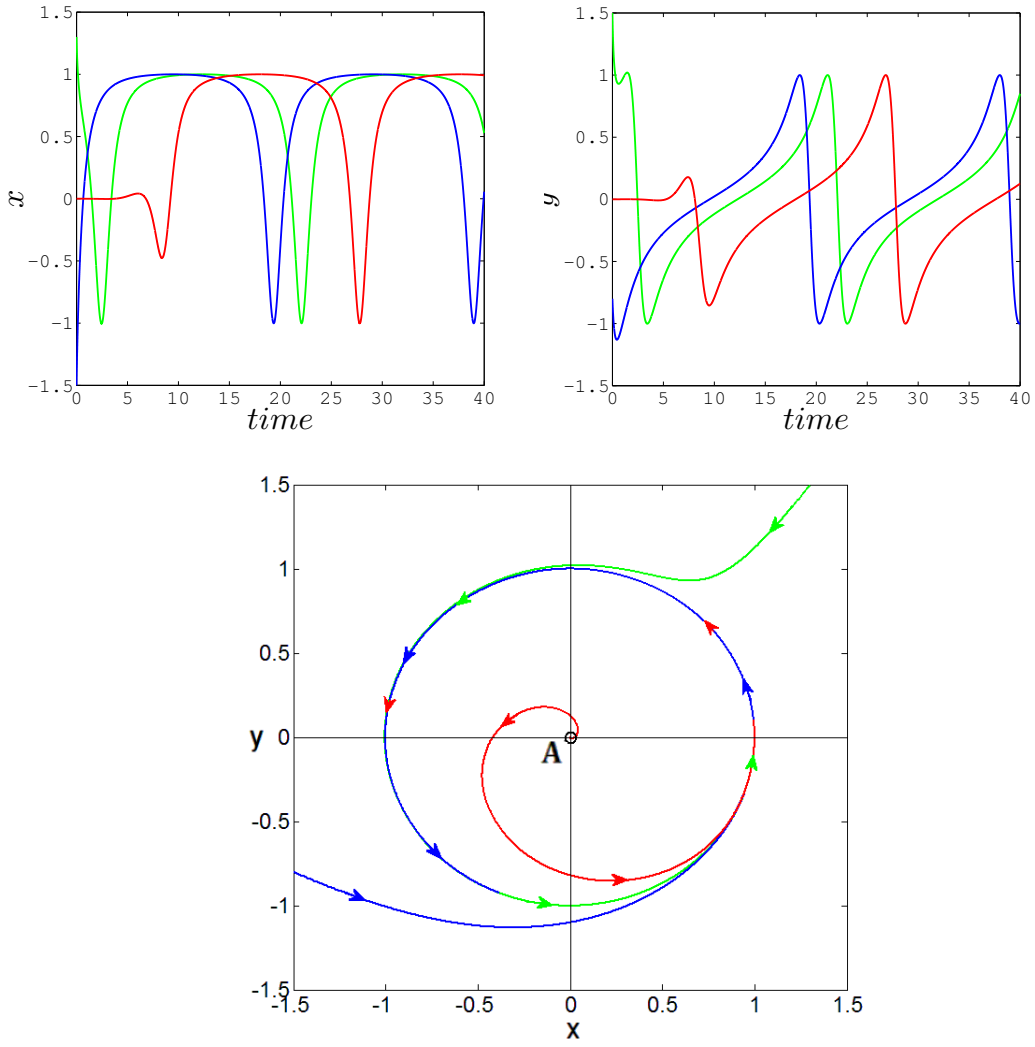

Fig. 2.3.: Time series and phase portrait above the bifurcation ($b > 1$): Point A denotes the unstable focus. Any trajectory oscillates counter clockwise.

3. FitzHugh-Nagumo Model

3.1. Simplified Model Equations

In order to grasp the complicated interaction between 10^9 of neurones in large neural networks, those are often lumped into groups of neural populations. Each of these populations can be represented as an effective excitable element that is coupled to the other elements. Since the modeling of large ensemble of neurones or neural populations is often challenging it can be helpful to use simple systems to describe the individual elements.

A second, generic model is given by the simplified FitzHugh-Nagumo systemsimplified FitzHugh-Nagumo system, which is often used as a paradigmatic generic model for this purpose and also for excitable systems in a general context:

$$\varepsilon \dot{x} = x - \frac{x^3}{3} - y \tag{3.1a}$$

$$\dot{y} = x + a, \tag{3.1b}$$

where x is called the activator, reproducing the behavior of the voltage during the course of a spike. y is called the inhibitor because it inhibits the production of x, while an increased production of x intensifies the production of y. Such an interplay of an activator and an inhibitor variable is typical for excitable systems. ϵ is a positive parameter that is usually chosen to be much smaller than unity because of timescale ratio of the activator and inhibitor variables. We will use $\epsilon = 0.005$ in the following. a is called the threshold parameter. We will see in section 3.2 that a determines whether the system is excitable ($a > 1$) or exhibits periodic firing (autonomous oscillations) ($a < 1$). Thus, a is a bifurcation parameter.

We call Eqs.(3.1) the simplified FitzHugh-Nagumo system, because there is also an extended version of the model that includes a linear y-dependence in the second equation. We will discuss that model later in section 3.4.

We have seen in the previous chapter that type-I excitability refers to a saddle-node infinite period bifurcation. There exists a second class of excitability, so called Type-II neural excitability, that is observed when a fixed point or resting potential loses stability via the supercritical Andronov-Hopf bifurcation. The emerging orbit has zero amplitude but non-zero frequency. Hence, the bifurcation results in a small-amplitude limit cycle . As the magnitude of the injected current increases, the amplitude of the limit cycle increases as well until it becomes a full-size spiking limit cycle.

3.2. Bifurcation Analysis

The set of points, where the vector field changes its horizontal and vertical direction is called the x-nullcline and y-nullcline, and it is defined by the equations:

$$0 = x - \frac{x^3}{3} - y \tag{3.2a}$$

$$0 = x + a. \tag{3.2b}$$

On these curves, the flow in phase space is purely horizontal and vertical, respectively. Indeed at any such point the dynamic variables x and y neither increase nor decrease because

$$\dot{x} = \dot{y} = 0 \tag{3.3}$$

and hence each point of intersection of the nullclines is an equilibrium point. This fixed point (x_A, y_A) of the neural system is given by

$$(x_A, y_A) = \left(-a, -a + \frac{a^3}{3}\right). \tag{3.4}$$

A linearization of Eqs. (3.1) around (x_A, y_A) yields for

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{1-a^2}{\varepsilon} & -\frac{1}{\varepsilon} \\ 1 & 0 \end{pmatrix}}_{=\mathbf{J}} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{3.5}$$

The eigenvalues Λ of the Jacobian matrix $\mathbf{J}$ determine the stability of the fixed point. They are given as roots of the characteristic equations

$$0 = \det \begin{pmatrix} \frac{1-a^2}{\varepsilon} - \Lambda & -\frac{1}{\varepsilon} \\ 1 & -\Lambda \end{pmatrix} \tag{3.6a}$$

$$= \Lambda^2 - \Lambda \,\mathrm{tr}\mathbf{J} + \det \mathbf{J} \tag{3.6b}$$

$$= \Lambda^2 - \frac{1-a^2}{\varepsilon}\Lambda + \frac{1}{\varepsilon} \tag{3.6c}$$

$$\Rightarrow \qquad \Lambda_{1,2} = \frac{1 - a^2 \pm \sqrt{(1-a^2)^2 - 4\varepsilon}}{2\varepsilon}. \tag{3.6d}$$

Since the timescale ratio ε is a positive constant, the determinant of $\mathbf{J}$ is also positive, i.e., $\det \mathbf{J} > 0$. Furthermore, the trace of the Jacobian matrix $\mathrm{tr}\mathbf{J} = (1 - a^2)/\varepsilon$ is positive if the absolute value of the parameter a is smaller than unity and negative for $|a| > 1$. Therefore, the fixed point is stable for $|a| > 1$, whereas for $|a| < 1$ the intersection of the nullclines occurs in the range of positive slope of the x-nullcline. Then the fixed point is unstable. In the nonlinear system, this change of stability happens by a Hopf bifurcation above which periodic oscillations exist. In the case of a stable fixed point, the system reaches the rest state (x_A, y_A) after an initial transient.

3.3. Numerical Simulations

After the discussion of the bifurcation scenarios, we will present results from numerical simulations in this section.

Below the bifurcation, $|a| > 1$, $(a = 1.01)$**:**

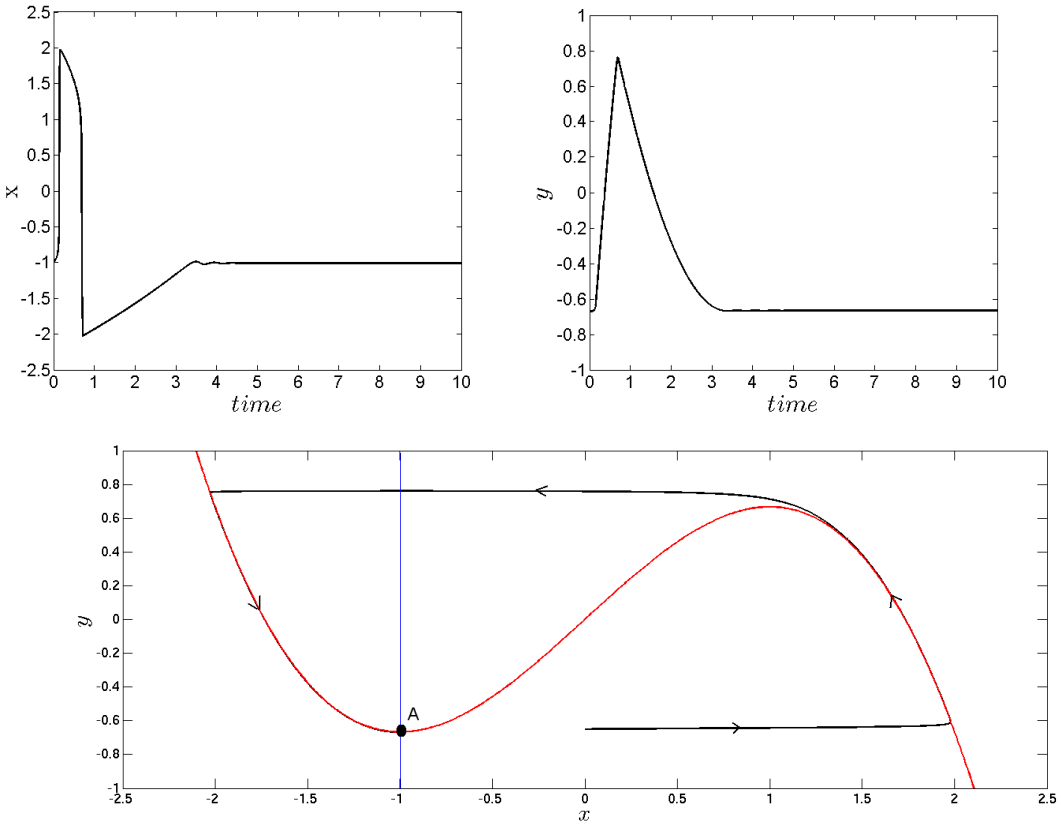

Fig. 3.1.: Time series and phase portrait of the FitzHugh-Nagumo system given by Eqs. (3.1) below the bifurcation for $a = 1.01$. The timescale ratio is chosen as $\varepsilon = 0.005$. The x- and y-nullclines are depicted by the red and blue curves, respectively. Point A denotes the fixed point.

Figure 3.1 displays the dynamics of FitzHugh-Nagumo system below the bifurcation, i.e., for $a = 1.01$. Upper panels depict the time series of the activator x and inhibitor y, respectively. Lower panel shows the trajectory in the (x, y)-phase space, where the nullclines are added for better orientation. The activator x-nullcline is shown as red curve and the blue line depicts the inhibitor y-nullcline. The arrows indicate the propagation direction of the trajectory. The intersection of the nullclines is the fixed point A.

An excitation cycle exhibits the following stages: Starting at the fixed point, the system performs subthreshold oscillations and is eventually lifted above the threshold by coupling to other elements or an external stimulus. Then the trajectory jumps horizontally to the distant branch of the x-nullcline. This rapid change of the activator variable x is due to the small timescale ratio ε. Excitations can be triggered by additional input that could arise from random fluctuations in the input signal or specific coupling functions describing the interaction with other elements.

Above the bifurcation, $|a| < 1, \quad (a = 0.97)$**:**

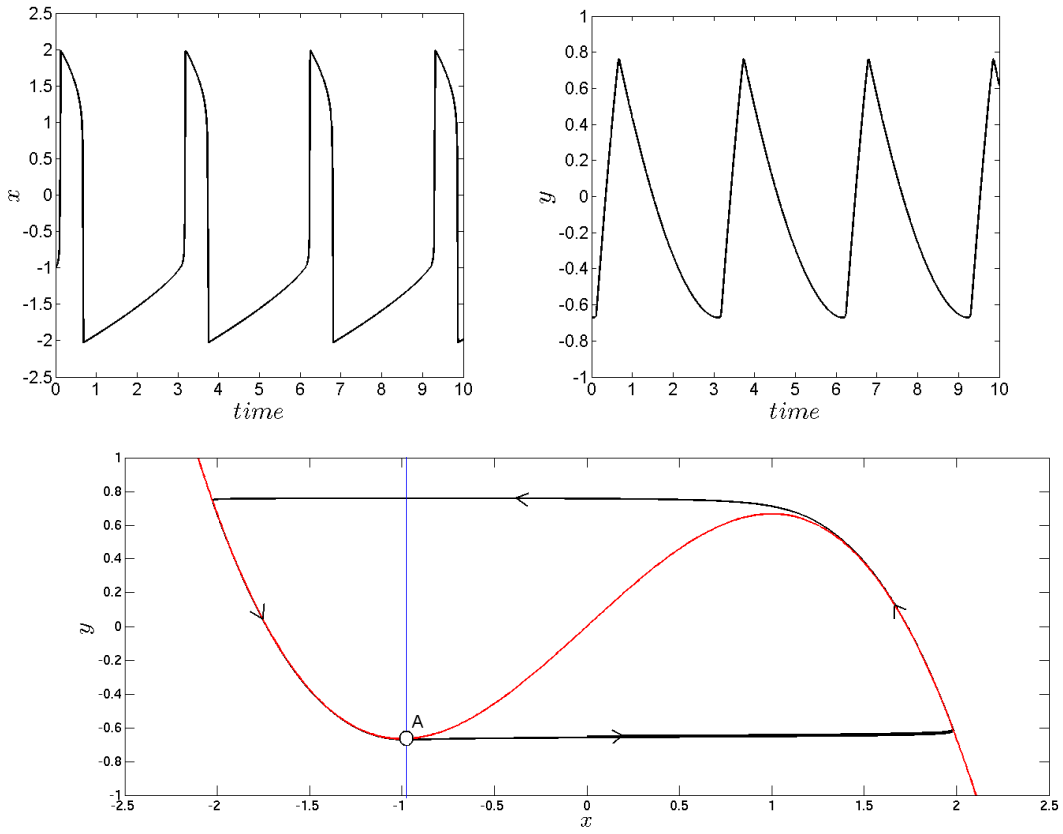

Fig. 3.2.: Time series and phase portrait of the FitzHugh-Nagumo system given by Eqs. (3.1) above the bifurcation for $a = 0.97$. The timescale ratio is chosen as $\varepsilon = 0.005$. A denotes the unstable fixed point.

Figure 3.2 displays the dynamics above the bifurcation for $a = 0.97$. In this oscillatory regime, there is no need for an additional input to trigger an excitation. Instead the FitzHugh-Nagumo system exhibits self-sustained oscillations.

Further information can be found in Refs. [FIT61, NAG62, HAU06, HOE09].

3.4. Extended FitzHugh-Nagumo Model

In the previous section we discussed the FitzHugh-Nagumo model, where we assumed a vertical nullcline for the inhibitor. Then the bifurcation point, which separates the excitable and inhibitory regimes, is only determined by the value of the threshold parameter a. In the following we discuss a slightly modified version of the FitzHugh-Nagumo system as presented in Eqs. (3.1). Its difference arises from an additional linear inhibitory term in the $\dot{y}$-equation:

$$\varepsilon \dot{x} = f(x,y) = x - \frac{x^3}{3} - y \tag{3.7a}$$

$$\dot{y} = g(x,y) = x + a - \gamma y \tag{3.7b}$$

with a new parameter $\gamma \in \mathbb{R}$. Now both γ and a determine the excitation threshold. We consider the case of positive γ, that is, a y-nullcline with a positive slope:

$$y = \frac{x}{\gamma} + \frac{a}{\gamma}. \tag{3.8}$$

Both the x-nullclines and y-nullclines are shown in Fig. 3.3. It becomes clear from Eq. (3.8) that a only shifts the y-nullcline in the y-direction, while γ also influences its slope. As a result they change the coordinates of the fixed point. The vertical y-nullcline of sections 3.1-3.3 is recovered in the limit $\gamma \to \infty$.

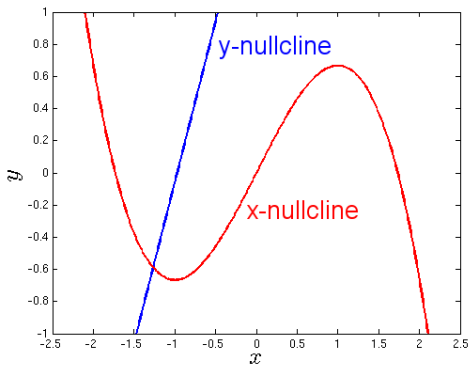

Fig. 3.3.: The red curve is the $x-$nullcline ($\dot{x} = 0$) and the blue line is the y-nullcline ($\dot{y} = 0$). Parameters: $a = 0.97$ and $\gamma = 0.5$.

In the following we explore how the linear stability changes for $\gamma \neq 0$. At first we need to determine the coordinates of the fixed point (x^*, y^*) of the system, that is the intersection of the nullclines:

$$y = x - \frac{x^3}{3} \quad \text{and} \quad y = \frac{x}{\gamma} + \frac{a}{\gamma}. \tag{3.9}$$

This is equivalent to finding the roots of the following polynomial of third order:

$$0 = -\frac{a}{\gamma} + \left(1 - \frac{1}{\gamma}\right) x - \frac{x^3}{3}. \tag{3.10}$$

We will show in later chapter how these roots can be found by analytical means. See section 4.2 for details.

In the next step we insert the coordinates in the linearization of Eqs (3.7):

$$\begin{pmatrix} \delta\dot{x} \\ \delta\dot{y} \end{pmatrix} \approx \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\delta y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\delta y} \end{pmatrix} \begin{pmatrix} \partial x \\ \partial y \end{pmatrix} = \begin{pmatrix} 1 - x^{*2} & -1 \\ \varepsilon & -\varepsilon\gamma \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \mathbf{J} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}. \tag{3.11}$$

Finally, the eigenvalues of $\mathbf{J}$ are calculated as follows:

$$\Lambda_{1,2} = \frac{\mathrm{tr}\mathbf{J} \pm \sqrt{\mathrm{tr}^2\mathbf{J} - 4\det\mathbf{J}}}{2}. \tag{3.12}$$

The real part of the eigenvalues indicates the stability of the fixed point, as discussed in the previous chapters. If both real parts of the eigenvalues are negative, then we have a stable fixed point. If they are positive, the fixed point is unstable. If they have different signs, there exists a saddle point. Moreover, if the eigenvalues are real, the fixed point is a stable or unstable node. In the other case, if they have a non-zero imaginary part, then the fixed point is called a focus.

In this section, if we choose $0 < \gamma < 1$ for our model, we find that the determinant is always positive:

$$\det\mathbf{J} = \varepsilon\left[\gamma\left(x^{*2} - 1\right) + 1\right] > 0 \tag{3.13}$$

which means that the fixed point cannot be a saddle. The specific type, stable/unstable node or focus, depends on the values of $\mathrm{tr}\mathbf{J}$ and $\det\mathbf{J}$. There are four possibilities:

1. $0 < 4\det\mathbf{J} < \mathrm{tr}^2\mathbf{J}$ and $0 < \mathrm{tr}\mathbf{J}$: both eigenvalues are real and larger than zero (unstable node).

2. $0 < 4\det\mathbf{J} < \mathrm{tr}^2\mathbf{J}$ and $\mathrm{tr}\mathbf{J} < 0$: both eigenvalues are real and smaller than zero (stable node).

3. $0 < \mathrm{tr}^2\mathbf{J} < 4\det\mathbf{J}$ and $0 < \mathrm{tr}\mathbf{J}$: there are two complex conjugate eigenvalues with real parts larger than zero (unstable focus).

4. $0 < \mathrm{tr}^2\mathbf{J} < 4\det\mathbf{J}$ and $\mathrm{tr}\mathbf{J} < 0$: there are two complex conjugate eigenvalues with real parts smaller than zero (stable focus).

The transition between stable and unstable foc (cases 3 and 4) occurs in a Hopf bifurcation, when the real parts become zero. Thus the eigenvalues cross the imaginary

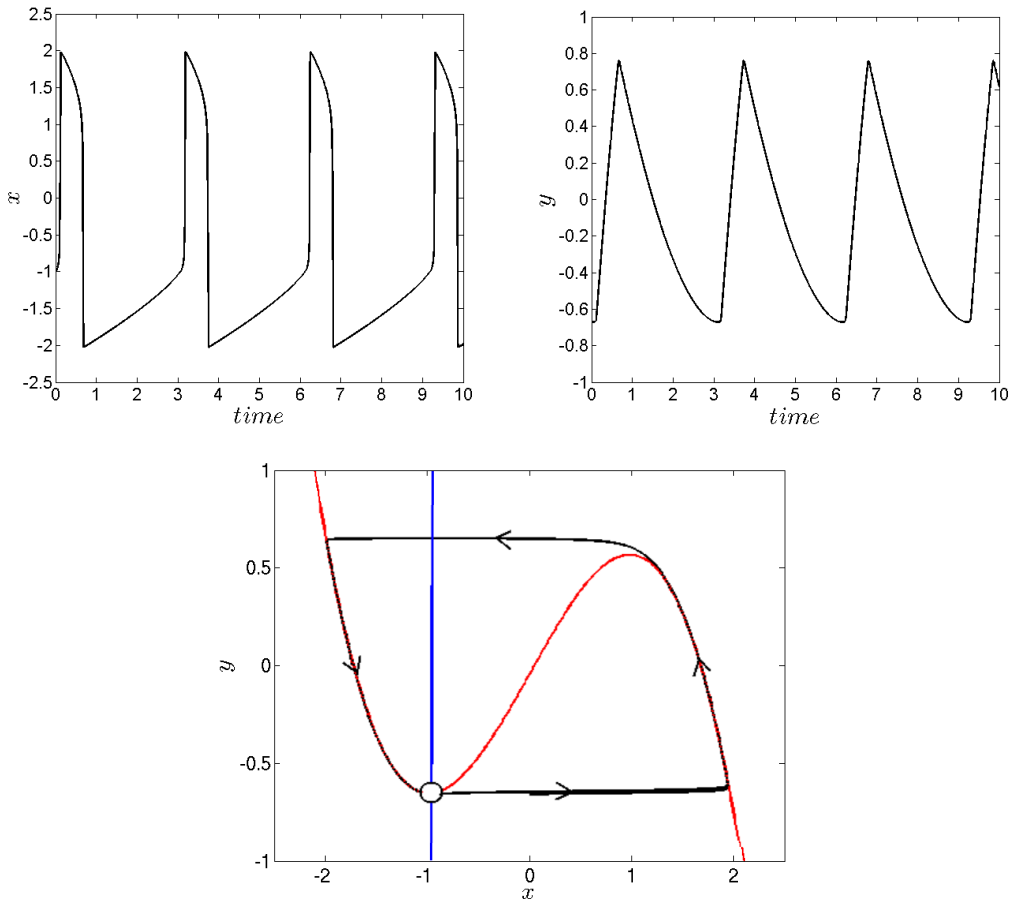

Fig. 3.4.: Time series and the phase portrait of the oscillatory regime. In the phase portrait the open circle marks the unstable fixed point. Parameters: $a = 0.97$, $\varepsilon = 0.005$, and $\gamma = 0.005$.

axis. This happens, when the trace of $\mathbf{J}$ becomes zero. This leads to a condition, which depends on the parameters ε, a, and γ:

$$\mathrm{tr}\mathbf{J} = 1 - x^*(a,\gamma)^2 - \varepsilon\gamma = 0. \tag{3.14}$$

Let us study the Hopf bifurcation more in detail for different parameter ranges. Figures 3.4 and 3.5 depict the dynamical scenarios in the excitable and oscillatory regime, where only the parameter γ is varied and the value of a remains fixed at $a = 0.97$. Compare these figures to Figs. 3.1 and 3.2, which show the case of the simple FitzHugh-Nagumo system.

We know that in the limit $\gamma \to \infty$ and for $|a| < 1$ the nullclines intersect on the unstable part of the cubic nullcline. For $a = 0.97$, this still holds for small values of γ as shown in Fig. 3.4, where the linear nullcline appears almost vertical. Accordingly the time series exhibit periodic oscillations. For larger $\gamma < 0$, the y-nullcline is

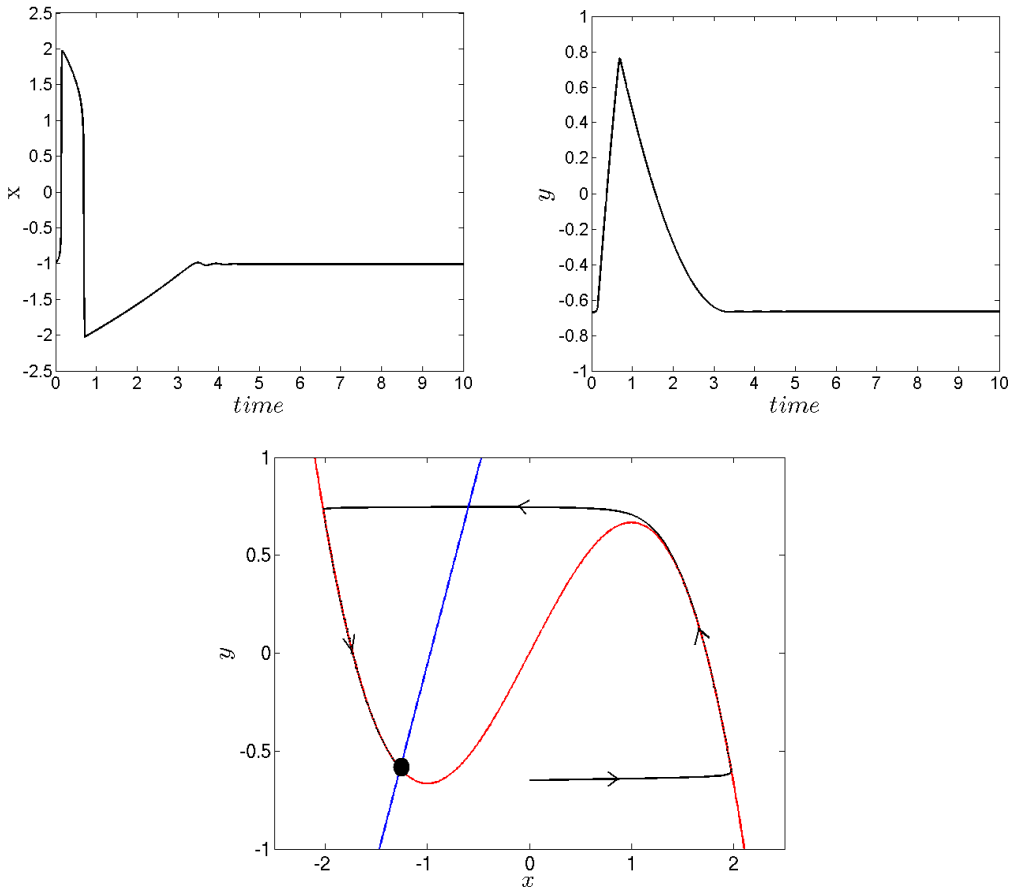

Fig. 3.5.: Time series and the phase portrait of the excitable regime. In the phase portrait the black dot indicates the stable fixed point. Parameters: $a = 0.97$, $\varepsilon = 0.005$, and $\gamma = 0.5$.

tilted such that the fixed point is now found to the left of the local minimum of the cubic nullcline. Thus the fixed point becomes stable and the system operates in the excitable regime. This is shown in Fig. 3.5.

Canard trajectory: There is also a trajectory shown in the Fig. 3.6 that runs though the relative maximum of the x-nullcline, and is called Canard trajectory. Above and below the Canard trajectory exist suprathreshold and subthreshold trajectories respectively. Canard trajectory marks a very sensible region in the dynamics of the system. If we have perturbations that drive the system above the Canard trajectory the system responds by an excitation cycle. Further information can be found in Ref. [SCH08e].

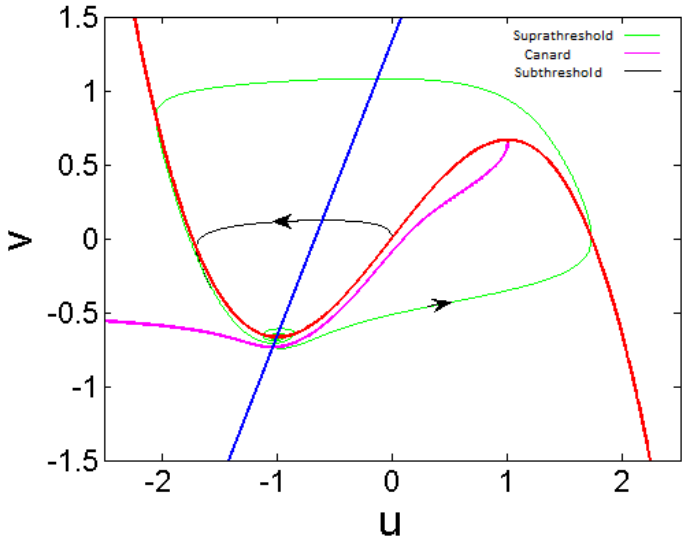

Fig. 3.6.: Phase portrait, and trajectories with subthreshold and suprathreshold initial conditions and Canard trajectory for an excitable system. $a = 0.67$ and $\gamma = 0.5$.

4. Hindmarsh-Rose Model

Besides phenomenological models, there is also a variety of models that are more closely related to physical quantities of neurological processes. Probably the most famous example of this class is the Hodgkin-Huxley model of the nerve impulse [HOD52]. As it will be explained later in section 7.1, this model consists of four coupled nonlinear differential equations involving six nonlinear functions and seven constants. The complexity of these equations demands numerical simulations and it is difficult to understand the fundamentals of the underlying nonlinear dynamics. An analytical treatment appears impossible for the investigations of interactions even in small neural networks. To address this purpose there is need for simple models that nevertheless exhibit the characteristic dynamical behavior of neurons.

4.1. Two-variable Hindmarsh-Rose Model

We have already met two systems in chapters 2 and 3, which despite their simple structure provide an insight into neural dynamics. The SNIPER model of chapter 2 captures the basic features of the excitability class I, where excitation occurs via a saddle-node bifurcation. At the bifurcation self-sustained oscillations emerge with zero frequency, i.e., infinite period, and finite amplitude. In addition chapter 3 introduces the FitzHugh-Nagumo system, which is governed by a Hopf bifurcation and thus models the excitability class II. There, oscillations start with zero amplitude at the bifurcation point and inherit the frequency from the timescale of a fixed point, which changes its stability.

In short both models are two-dimensional dynamical systems corresponding to different frequency-current relationships. Formally they could be written as an activator-inhibitor system of the form

$$\dot{x} = f(x, y) \tag{4.1a}$$
$$\dot{y} = g(x, y), \tag{4.1b}$$

where x and y denote the activator and inhibitor, respectively, and f and g are potentially nonlinear functions specifying the dynamics.

In this chapter we will focus on another two-dimensional model that does not only combine both types of bifurcation within one set of equations. It also exhibits additional codimension-two bifurcations, which involve two independent bifurcation parameters. Specifically, the two-dimensional Hindmarsh-Rose model takes the following form [TSU07]:

$$\dot{x} = c\left(x - \frac{x^3}{3} - y + z\right) \tag{4.2a}$$
$$\dot{y} = \frac{x^2 + dx - by + a}{c}, \tag{4.2b}$$

where a and d are bifurcation parameters. For simplicity, we will keep the other parameters fixed at $b = 1$, $c = 3$, and $z = 0$. Originally the Hindmarsh-Rose model involves 3 variables that operate each on different timescales [HIN82, HIN84]. We will discuss this extended model and the reductions to Eqs. (4.2) in section 4.5.

Equations (4.2) consist of a cubic nonlinearity in the activator equation as in FitzHugh-Nagumo model, but the equation of the inhibitor involves now a quadratic nonlinearity.

We will see in the following that this system may, in general, exhibit typical nonlinear phenomena such as Andronov-Hopf and saddle-node bifurcations, separatrix loops, and other dynamical features depending on the bifurcation parameters. These are given by the parameters a and d.

4.2. Analytical Solutions for the Roots of a Cubic Polynomial

In this section we will introduce the analytical solutions of a cubic function. This allows for calculation of fixed points of the Hindmarsh-Rose Eqs. (4.2), which are given by the intersections of the nullclines for the activator and the inhibitor. We will then use the coordinates of the fixed points to determine their stability by calculating the eigenvalues of the Jacobian matrix.

In general a cubic equation, or in other words a polynomial of degree three can be written as:

$$\alpha x^3 + \beta x^2 + \gamma x + \delta = 0. \tag{4.3}$$

The *discriminant* Δ of Eq. (4.3) is given by:

$$\Delta = \beta^2\gamma^2 - 4\alpha\gamma^3 - 4\beta^3\delta - 27\alpha^2\delta^2 + 18\alpha\beta\gamma\delta. \tag{4.4}$$

This quantity is helpful to calculate the number of solutions of Eq. (4.3). In fact, their number is determined by the following cases:

1. $\Delta > 0$: three distinct real roots.
2. $\Delta = 0$: one root of multiplicity 2 and a second real root.
3. $\Delta < 0$: one real root and two complex conjugate roots with nonzero imaginary parts.

In the following we will apply this general finding to the calculation of the fixed points of the two-dimensional Hindmarsh-Rose model given by Eqs. (4.2). Then we will subsequently discuss the stability of each fixed point.

The fixed points (x_i^*, y_i^*) of Eqs. (4.2) with $b = 1$ and $c = 3$ are the solutions of the following system of equations:

$$3\left(x - \frac{x^3}{3} - y\right) = 0 \tag{4.5a}$$

$$\frac{x^2 + dx - y + a}{3} = 0. \tag{4.5b}$$

In order to analytically solve this set of equations, it is helpful to first find the discriminant Δ of the equation. For simplicity, we fix the parameters $b = 1$, $c = 3$, and $z = 0$ as mentioned above. First we obtain y in dependence on x via Eq. (4.5a):

$$y = x - \frac{x^3}{3}. \tag{4.6}$$

This expression can be used to rewrite Eq. (4.5b), which yields

$$x^2 + dx - \left(x - \frac{x^3}{3}\right) + a = 0 \tag{4.7a}$$

$$\Leftrightarrow \qquad x^3 + 3x^2 + 3(d-1)x + 3a = 0. \tag{4.7b}$$

Comparing Eq. (4.7b) to Eqs. (4.3) and (4.4), we find

$$\alpha = 1, \beta = 3, \gamma = 3(d-1), \text{ and } \delta = 3a. \tag{4.8}$$

One can now calculate the discriminant Δ via Eq. (4.4) in dependence on the parameters a and d:

$$\Delta(d, a) = 81(d-1)^2 - 108(d-1)^3 - 324a - 243a^2 + 486(d-1)a. \tag{4.9}$$

The result is shown in Fig. 4.1 as a two-dimensional projection onto the (a, d)-plane. Warm (cold) colors indicate positive (negative) values of Δ. The black curve marks the threshold case $\Delta = 0$. Inside this curve, we find that Δ is smaller than zero. In this case, Eq. (4.7b) has one real root and two complex conjugate roots. Outside this area, we have $\Delta > 0$ and find three distinct real solution.

Alternatively, we can also solve the cubic function in Eq. (4.3) in general using a couple of substitutions and then we apply the results to the equations of the two-dimensional Hindmarsh-Rose model: Dividing Eq. (4.3) by α and substituting x by $t - \frac{\beta}{3\alpha}$ we can eliminate the second-order term x^2. Thus, we obtain the following equation:

$$t^3 + pt + q = 0 \tag{4.10}$$

with the abbreviations $p = \frac{3\alpha\gamma - \beta^2}{3\alpha^2}$ and $q = \frac{3\beta^2 - 9\alpha\beta\gamma + 27\alpha^2\gamma}{27\alpha^3}$. As a side remark, the second-largest order term of any polynomial can always be eliminated by such a transformation.

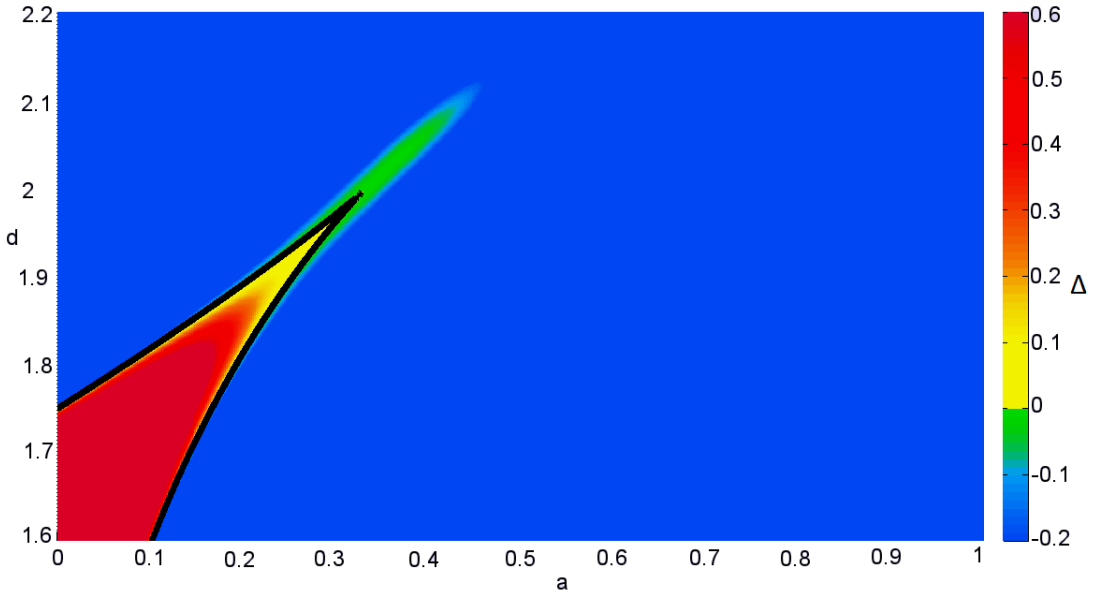

Fig. 4.1.: Projection of the discriminant Δ on the (a, d)-plane according to Eq. (4.4). The black curve s a contour line for $\Delta = 0$. Parameters: $\alpha = 1, \beta = 3, \gamma = 3(d-1)$, and $\delta = 3a$.

Now, the solutions of Eq. (4.10) can be found by using Cardano's method. We first apply the *depressed cubic* Eq. (4.10) by using two auxiliary variables u and v, which are linked by the condition $u + v = t$, and substitute this in our depressed Eq. (4.10). Thus we obtain:

$$u^3 + v^3 + (3uv + p)(u + v) + q = 0. \tag{4.11}$$

At this point we apply the second condition of Cardano's method: $3uv + p = 0$ or equivalently $u^3v^3 = -p^3/27$. Since one term in Eqs. (4.11) vanishes, we get: $u^3 + v^3 = -q$. Using the substitution $w = u^3$, we arrive at the equation

$$w^2 + qw - \frac{p^3}{27} = 0. \tag{4.12}$$

Therefore a quadratic equation is obtained, for which we can easily find the roots w. Then we subsequently determine the values for u and v as well as t and finally the initially desired solution x.
Considering $a = 0.6$ and $d = 2$, for example, we obtain the following real fixed point:

$$(x^*, y^*) = (-1.93, 0.46) \tag{4.13}$$

and two additional complex roots. Compare Fig. 4.1 where this choice of parameters yields $\Delta < 0$. Next we insert the real root (x^*, y^*) in Jacobian matrix of system (4.2),

which is given by

$$\mathbf{J} = \begin{pmatrix} c - cx^2 & -c \\ \frac{2}{c}x + \frac{d}{c} & -\frac{b}{3} \end{pmatrix}. \tag{4.14}$$

Thus, we obtain for our choice of parameters ($a = 0.6$, $b = 1$, $c = 3$, and $d = 2$)

$$\mathbf{J} = \begin{pmatrix} 3 - 3(x_1^*)^2 & -3 \\ \frac{2}{3}x_1^* + \frac{2}{3} & -\frac{1}{3} \end{pmatrix}. \tag{4.15}$$

In order to calculate the eigenvalues we need to solve the characteristic equation:

$$\det(\mathbf{J} - \Lambda\,\mathbf{I}) = 0 \tag{4.16a}$$

$$\Leftrightarrow \quad \Lambda^2 - \Lambda\ \mathrm{tr}\mathbf{J} + \det\mathbf{J} = 0, \tag{4.16b}$$

where Λ denotes the eigenvalues, tr refers to the trace, and $\det$ is the determinant of the Jacobian matrix $\mathbf{J}$. $\mathbf{I}$ names the identity matrix. Inserting the coordinates of the fixed points in $\mathbf{J}$ leads to the respective eigenvalues.

Inserting the coordinates of the first fixed point $(x^*, y^*) = (-1.93, 0.46)$ in Jacobian matrix (4.14) and solving characteristic equation (Eqs. (4.16) yields the eigenvalues:

$$\Lambda_1 = -0.10 \tag{4.17a}$$

$$\Lambda_2 = -8.39. \tag{4.17b}$$

Both eigenvalues Λ_1 and Λ_2 are smaller than zero. Therefore the fixed point located at $(-1.93, 0.46)$ is a stable node. This is in agreement with Fig. 4.2. Keep in mind that the bifurcating parameters, which we choose for our exemplary calculation ($a = 0.6$ and $d = 2$), are located in region (a). The stable node is shown in the schematic diagram Fig. 4.3. Compare also Tab. 4.1.

4.3. Bifurcations in Two-Dimensional Hindmarsh-Rose Model

In this section we will provide the general bifurcation diagram of the two-dimensional Hindmarsh-Rose model given by Eqs. (4.2). Details such as time series illustration and different bifurcations will be given in section 4.4.

Figure 4.2 shows a bifurcation diagram of limit sets of the solutions in Eqs. (4.2) in the (a, d) parameter plane. In this figure, the notation defined in Refs. [HOP97, TSU07] are used. Let us start exploring the bifurcation diagram with the yellow region (c), where there exist an unstable fixed point and a stable limit cycle. From chapter 3 we know that a limit cycle becomes smaller and smaller as it approaches a Hopf bifurcation, where it vanishes and transfers its stability to the fixed point. This scenario can also be found in the two-dimensional Hindmarsh-Rose model and

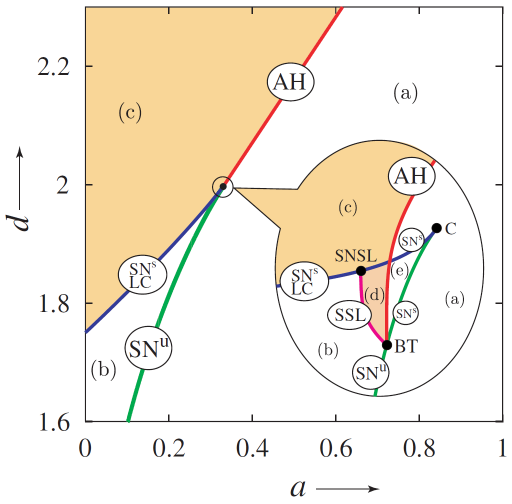

Fig. 4.2.: Bifurcation diagram of the two-dimensional Hindmarsh-Rose model given by Eqs. (4.2) in the (a, d) parameter plane for $b = 1$ and $c = 3$. AH: Andronov-Hopf bifurcation, SN: saddle-node bifurcation, SNLC: saddle-node on limit cycle bifurcation, BT: Bogdanov-Takens bifurcation, C: cusp bifurcation, SNSL: saddle-node on separatrix loop bifurcation, SSL: saddle-separatrix loop bifurcation. For details see text. The figure is a reproduction of Fig. 1 in Ref. [TSU07].

is indicated by the red curve labeled AH for an Andronov-Hopf bifurcation. To the right of this curve, in region (a), there exist only a stable fixed point. This is the case discussed at the end of the previous section.

The limit cycle present in region (c) can also undergo a saddle-node bifurcation on a limit cycle also known as saddle-node infinite period bifurcation (SNIPER). Compare also chapter 2 for details on this bifurcation. In fact this bifurcation is also possible in Eqs. (4.2) for smaller values of the parameter a. See the blue curve in Fig. 4.2 marked as SNLC corresponding to the bifurcation just mentioned. Beyond this bifurcation, in region (b), the limit cycle does not exist anymore and the set of invariant solutions consists of a saddle, a stable and an unstable fixed point.

Increasing the parameter a the saddle and the unstable fixed point can collide and annihilate without forming a limit cycle as indicated by the green curve labeled SN for saddle-node bifurcation. Then one has reached again in the region (a) in the bifurcation with a single stable fixed point.

The red, blue, and green bifurcation curves seem to meet in a single point, but the enlargement in Fig. 4.2 shows that they collide only in pairs. These points refer to codimension-2 bifurcations, The green SN and red AH curves meet in a so-called Bogdanov-Takens (BT) bifurcation. When the blue SNLC curve and the green SN line coexist for the same parameter combination (a, d), we have a cusp bifurcation. See point C in the inset of Fig. 4.2. In addition there exists a saddle-separatrix loop

Label	Description
(a)	1 stable fixed point
(b)	1 saddle, 1 stable, 1 unstable fixed point
(c)	1 unstable fixed point, 1 stable limit cycle
(d)	1 saddle, 1 stable fixed points, 1 unstable fixed point, 1 stable limit cycle
(e)	1 saddle, 2 stable fixed points
AH	(Andronov-)Hopf bifurcation
SNLC	Saddle-node bifurcation on a limit cycle
SN	Saddle-node bifurcation (of equilibria)
BT	Bogdanov-Takens bifurcation
C	Cusp bifurcation
SSN	Saddle-separatrix loop bifurcation
SNSL	Saddle-node on separatrix loop bifurcation

Tab. 4.1.: Summary of regions and bifurcations displayed in Fig. 4.2 .

(SSL) bifurcation connecting the green and blue curves for smaller parameters a. It bounds the region (d) with a stable limit cycle next to the saddle and the two fixed points of regions (b) and ends in a saddle-node on separatrix loop bifurcation marked as SNSL on the blue SN curve. Furthermore, the stable limit cycle eventually undergoes a Hopf bifurcation at the red boundary of region (d), where it merges with the unstable fixed point. Thus, in region (e) bounded by the red, blue, and green curves, there exist two stable fixed points and a saddle.

For later reference the different bifurcations and dynamical regions are summarized in Tab. 4.1.

4.4. Details on the Bifurcation Diagram

This section provides further explanations of the bifurcation diagram 4.2. In the reproduction of in Fig. 4.3 we also include schematic plots of the fixed points and limit cycles involved in the different bifurcations. Here are short summaries of the bifurcations mentioned in Fig. 4.2 and Tab. 4.1:

Andronov-Hopf: Andronov-Hopf bifurcation describes the birth of a limit cycle from an equilibrium in dynamical systems generated by ordinary differential equations (ODEs), when the equilibrium changes stability via a pair of purely imaginary eigenvalues. See also chapter 3, where this bifurcation appeared in the FitzHugh-Nagumo model.

Saddle-node: A saddle node bifurcation is a local bifurcation in which two fixed points (or equilibria) of a dynamical system collide and annihilate each other. The term *saddle-node bifurcation* is most often used in reference to continuous dynamical systems.

Saddle-node on a limit cycle: This bifurcation, which is also known as SNIPER

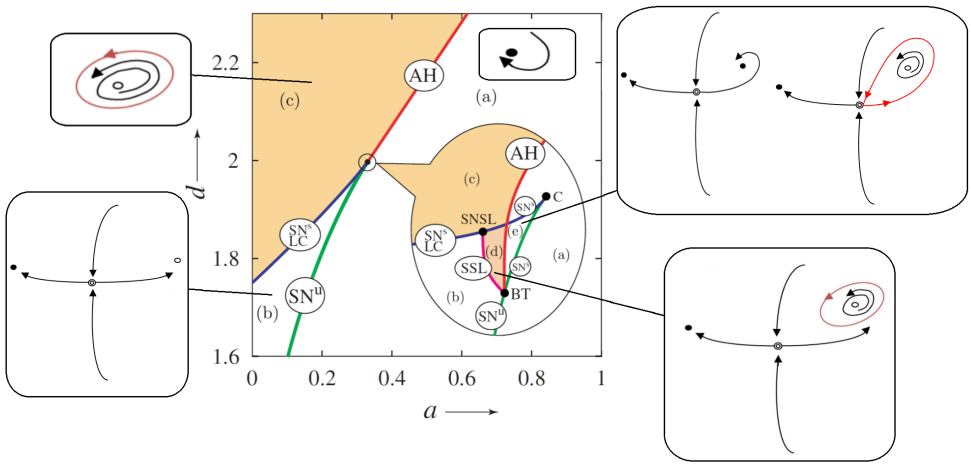

Fig. 4.3.: Schematic diagrams related to Tab. 4.1. In the schematic insets, black, white, and double circles represent stable, unstable, and saddle equilibrium points, respectively. Red closed curve denotes a stable limit cycle. Arrows indicate directions of trajectories.

(saddle-node infinite period bifurcation) or SNIC (saddle-node bifurcation on invariant circle) bifurcation, occurs when the center manifold of a saddle-node bifurcation forms an invariant circle. Such a bifurcation results in (dis)appearance of a limit cycle of an infinite period. See also section 2.2, where this bifurcation was a characteristic feature of the SNIPER model.

Fold-Hopf: The Jacobian matrix at the equilibrium has a pair of purely imaginary complex-conjugate eigenvalues (Andronov-Hopf bifurcation) and one zero eigenvalue (saddle-node bifurcation).

Bogdanov-Takens: The Jacobian matrix has two zero eigenvalues. a Bogdanov-Takens bifurcation is a well-studied example of a bifurcation with co-dimension two, meaning that two parameters must be varied for the bifurcation to occur.

Cusp: This is a bifurcation of equilibria in a two-parameter family of autonomous ODEs at which the critical equilibrium has one zero eigenvalue and the quadratic coefficient for the saddle-node bifurcation vanishes.

In the following we will discuss the difference regions in Fig. 4.2 in more detail and provide time series visualizing different initial conditions.

Bifurcation region (a): Figure 4.4 displays the dynamics of Hindmarsh-Rose system for a relatively wide range of bifurcating parameters a and d. Upper panels show the time series of the activator x and inhibitor y. Lower panel shows the trajectory in the (x, y)-phase space, and the x-nullcline and y-nullcline in red and blue curves, respectively. The stable fixed point is depicted as black dot in the figure.

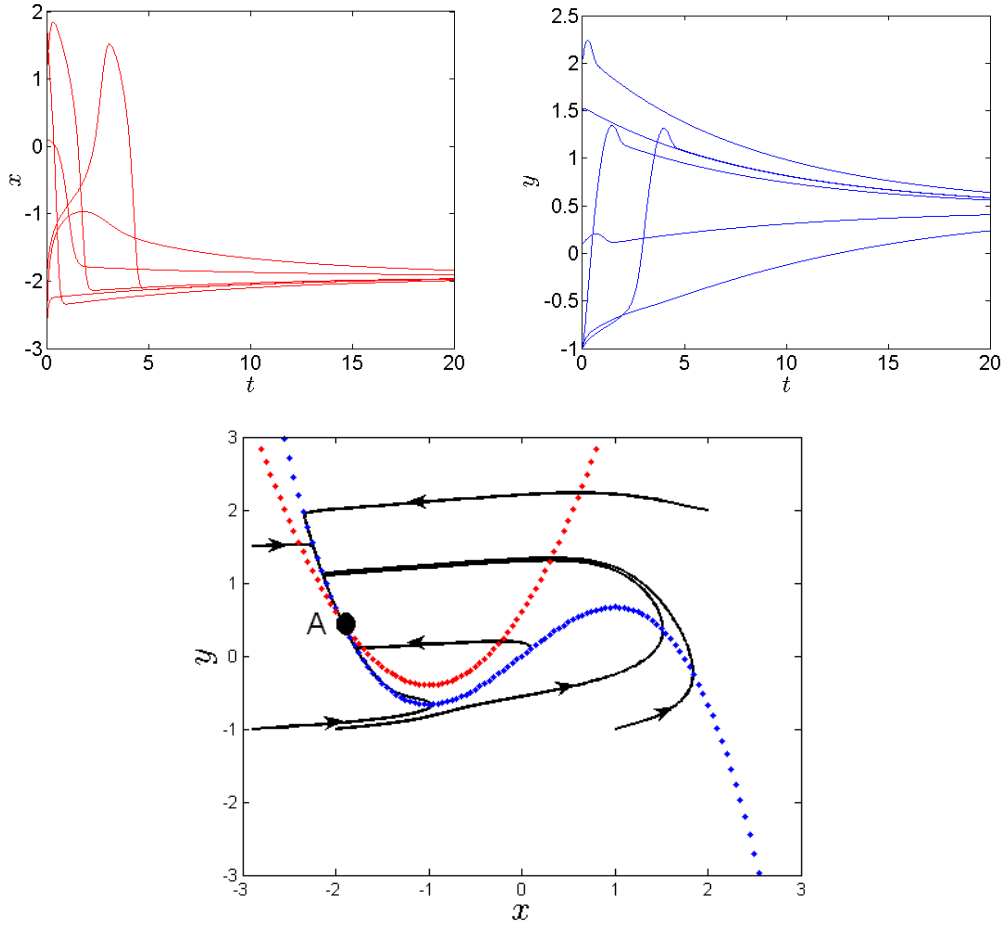

Fig. 4.4.: Time series and phase portrait of the region (a) in Figs. 4.2 and 4.3. Parameters: $a = 0.6$ and $d = 2$.

Following the derivations of Sec. 4.2 and solving the differential equations, for $a = 0.6$ and $d = 2$ corresponding region (a), we obtain the coordinates of the stable

node and by solving the characteristic equation (4.16), we obtain the eigenvalues:

$$(x_A^*, y_A^*) = (-1.93, 0.46), \tag{4.18a}$$
$$\Lambda_1 = -0.10, \tag{4.18b}$$
$$\Lambda_2 = -8.39. \tag{4.18c}$$

In this region we obtain $\mathrm{tr}\mathbf{J} < 0$, $\det \mathbf{J} > 0$, and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$. Thus the fixed point is a stable node. This is also in agreement with a negative discriminant Δ: $\Delta(a =, a = 0.6) = -17.28$

Bifurcation region (b): Figure 4.5 shows the time series of the system in upper panels and the dynamics of the system in the phase portrait for $a = 0.05$ and $d = 1.7$. This yields $\Delta = 2.48 > 0$ and we expect three distinct fixed points. In fact, Fig. 4.5 depicts a stable fixed point (black dot), a saddle point (double circles), and an unstable fixed point (empty dot) for this region.

In analogy to the linear stability investigations for region (a), we can follow the derivations of section 4.2, where we now change the parameters to $a = 0.05$ and $d = 1.7$ for region (b). We find the coordinates of the fixed points and subsequently the corresponding eigenvalues from the characteristic equation (4.16).

For the first fixed point we obtain

$$(x_A^*, y_A^*) = (-1.98, 0.59), \tag{4.19}$$
$$\Lambda_1 = -0.07, \tag{4.20}$$
$$\Lambda_2 = -8.97. \tag{4.21}$$

This yields $\mathrm{tr}\mathbf{J} < 0$, $\det \mathbf{J} > 0$, and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$ and this fixed point is a stable node (black dot in Fig. 4.5). Similarly, we find a second fixed point with

$$(x_B^*, y_B^*) = (-0.08, -0.08), \tag{4.22}$$
$$\Lambda_1 = 2.42, \tag{4.23}$$
$$\Lambda_2 = 0.23, \tag{4.24}$$

which leads to $\mathrm{tr}\mathbf{J} > 0$, $\det \mathbf{J} > 0$, and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$. This characterizes the fixed point as an unstable node (empty dot in Fig. 4.5). Finally, the third fixed point gives

$$(x_C^*, y_C^*) = (-0.94, -0.66), \tag{4.25}$$
$$\Lambda_1 = 0.54, \tag{4.26}$$
$$\Lambda_2 = -0.55. \tag{4.27}$$

Since we have $\det \mathbf{J} < 0$, this fixed point is a saddle node (double circle in Fig. 4.5).

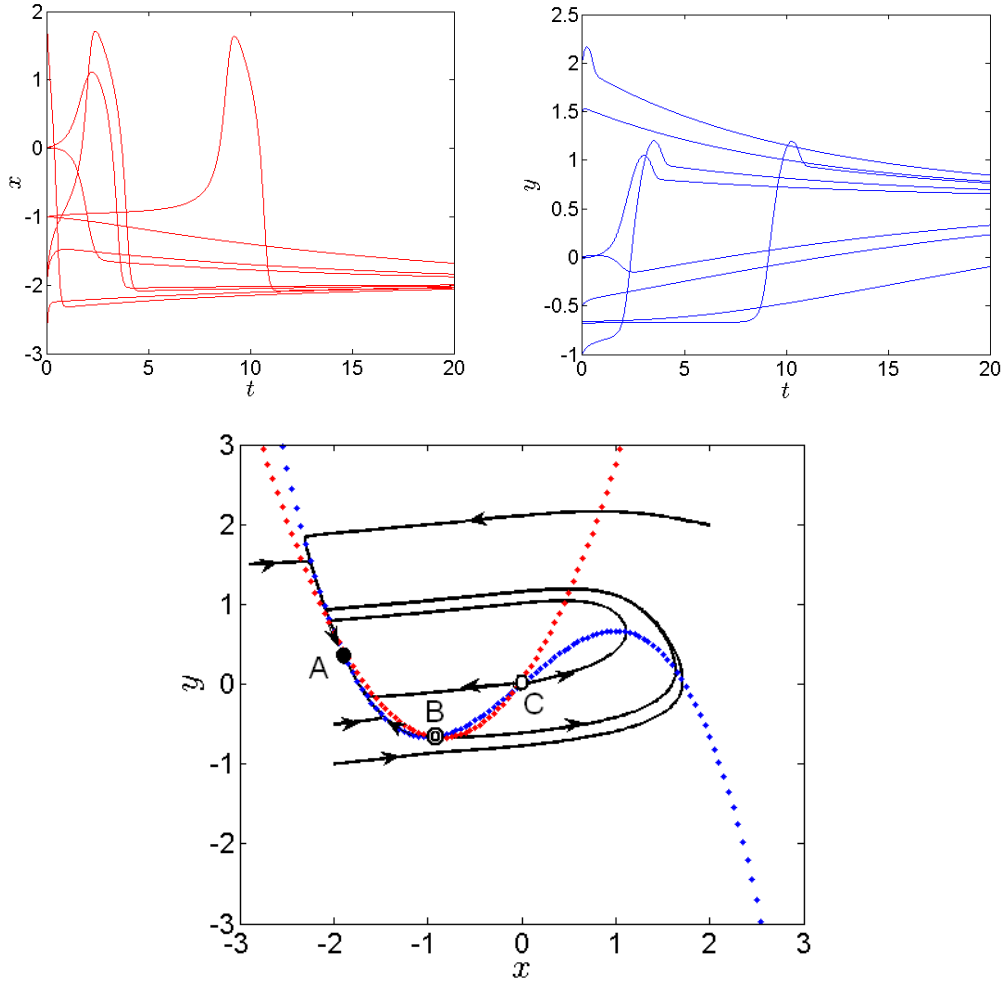

Fig. 4.5.: Time series and phase portrait of the region (b) in Figs. 4.2 and 4.3. Parameters: $a = 0.05$ and $d = 1.7$.

Bifurcation region (c): Figure 4.6 depicts an unstable fixed point (empty dot) and a stable limit cycle (red cycle), as shown in the picture a periodic activity is obtained in the region (c). It rapidly changes to a large limit cycle when the state of this system moves away from the Andronov-Hopf bifurcation curve (see Tab. 4.1). This phenomenon is often called a *Canard explosion*.

Following again the derivations of section 4.2 for $a = 0.2$ and $d = 1.9$ corresponding region (c), we obtain the coordinates of the stable node and by solving the characteristic equation (4.16) its eigenvalues:

$$(x_A^*, y_A^*) = (-0.13, -0.13), \tag{4.28a}$$

$$\Lambda_1 = 2.33, \tag{4.28b}$$

$$\Lambda_2 = 0.28. \tag{4.28c}$$

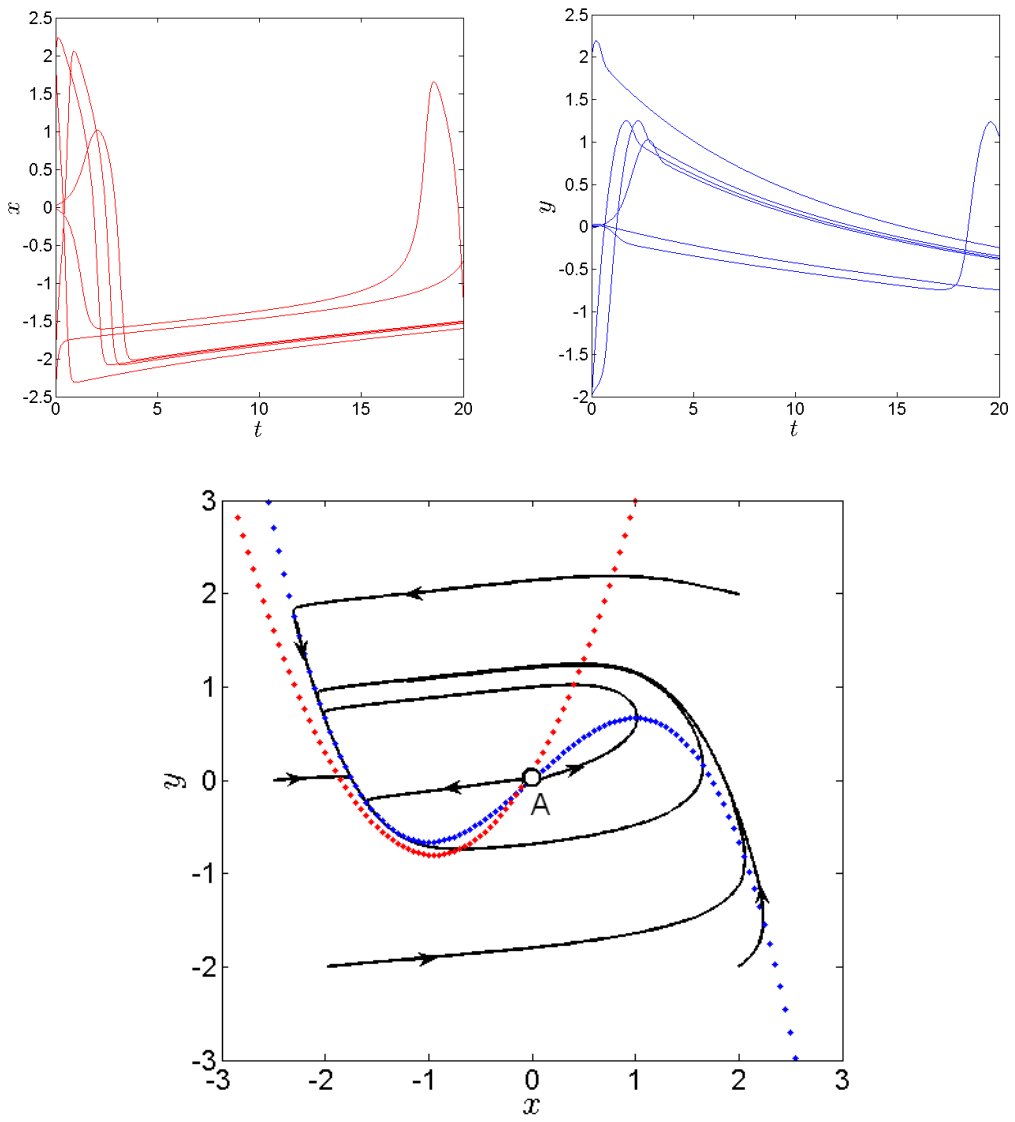

Fig. 4.6.: Time series and phase portrait of the region (c) in Figs. 4.2 and 4.3. Parameters: $a = 0.1$ and $d = 1.9$.

This yields $\mathrm{tr}\mathbf{J} > 0$, $\det \mathbf{J} > 0$, and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$ and we find that this fixed point is an unstable node, indeed. The existence of only one fixed point is confirmed by $\Delta = -0.16 < 0$.

Bifurcation region (d): In Fig. 4.7 there exist a stable fixed point (black dot), an unstable fixed point (empty dot), a saddle (double circles) and a stable limit cycle. A sink (stable fixed point) and a saddle that exist in region (d) disappear by SN^s (Tab. 4.1) in region (c). A small stable limit cycle exists just above the SN^s

bifurcation, and moreover it exists in the very narrow area along the AH bifurcation curve (see Tab. 4.1). In order to see all fixed points in the Fig 4.7 , it should be zoomed in several times.

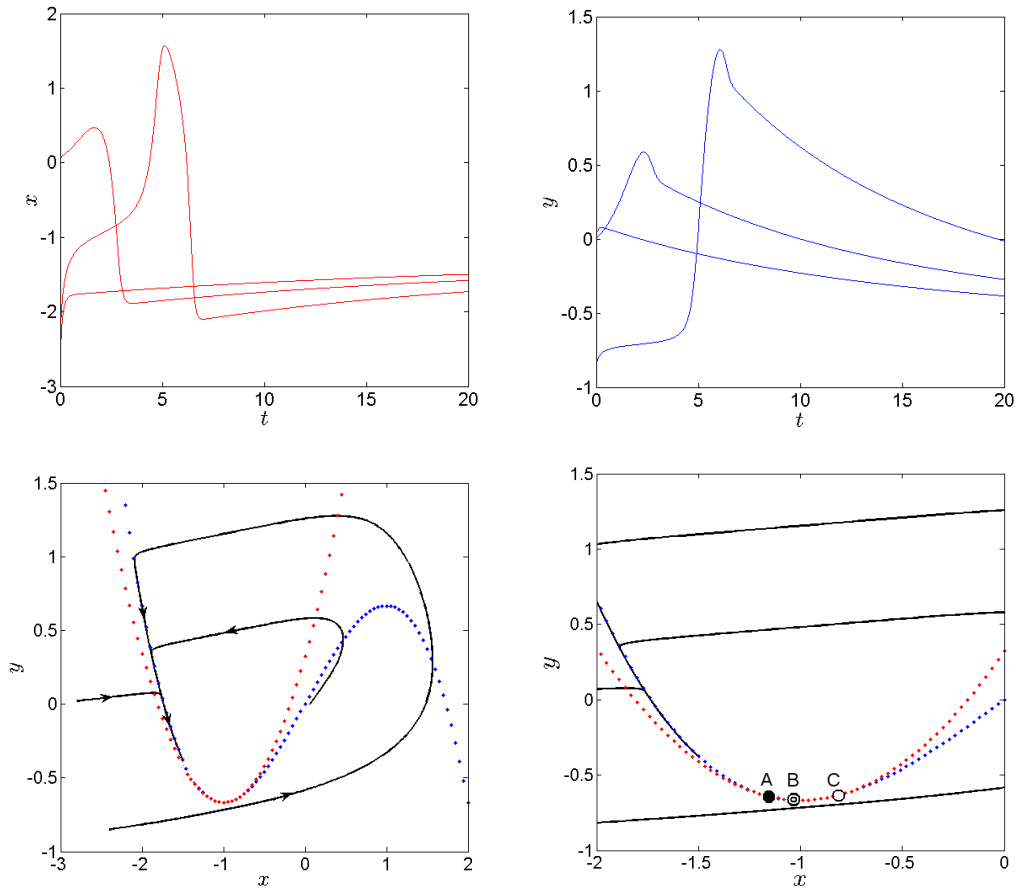

Fig. 4.7.: Time series and phase portrait of the region (d) in Figs. 4.2 and 4.3. The lower right panel is an enlargment of the phase portrait to show all three fixed points. Parameters: $a = 0.323$ and $d = 1.99$.

In analogy to the linear stability investigations for regions (a),(b) and (c), we follow the derivations of Sec. 4.2 once more. This time we change the parameters to $a = 0.323$ and $d = 1.99$ for region (d). For these parameters we obtain a small, but positive discriminant $\Delta = 0.000081$ indicating the existence of three distinct fixed points. Accordingly we find the coordinates of the fixed points and the corresponding eigenvalues as follows:

$$(x_A^*, y_A^*) = (-1.16, -0.64), \tag{4.29}$$

$$\Lambda_1 = -0.010, \tag{4.30}$$

$$\Lambda_2 = -1.34. \tag{4.31}$$

This yields $\mathrm{tr}\mathbf{J} < 0$, $\det \mathbf{J} > 0$ and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$ and we see that the fixed point is a stable node (black dot in Fig. 4.7). The second fixed point is found at

$$(x_B^*, y_B^*) = (-0.81, -0.63), \tag{4.32}$$
$$\Lambda_1 = 0.65, \tag{4.33}$$
$$\Lambda_2 = 0.04. \tag{4.34}$$

We see from $\mathrm{tr}\mathbf{J} > 0$, $\det \mathbf{J} > 0$, and $\mathrm{tr}^2\mathbf{J} > 4 \det \mathbf{J}$ the the fixed point is an unstable node (empty dot in Fig. 4.7). The third fixed point yields

$$(x_C^*, y_C^*) = (-1.03, -0.67), \tag{4.35}$$
$$\Lambda_1 = 0.02, \tag{4.36}$$
$$\Lambda_2 = -0.55 \tag{4.37}$$

and we have $\det \mathbf{J} < 0$. Thus, the fixed point is a saddle node (double circles in Fig. 4.7).

Further information about this two-dimensional model can also be found in Refs. [TSU07].

4.5. Three-dimensional Hindmarsh-Rose Model

In 1984, J. L. Hindmarsh and R. M. Rose extended their two years earlier introduced model, which consisted of two dynamical variables [HIN82], by a third variable. The third equation gives to burstingbursting activities, that is, a series of rapid oscillations followed by a recovery process. In total, this three-dimensional Hindmarsh-Rose model can be writte as follows (see Eq. (15) in Ref. [HIN84]):

$$\dot{x} = y - ax^3 + bx^2 - z + I \tag{4.38a}$$
$$\dot{y} = c - dx^2 - y \tag{4.38b}$$
$$\dot{z} = \varepsilon(s(x - x_0) - z), \tag{4.38c}$$

where x is known as membrane poential and y and z are known as fast and slow gating variables for ionic currents respectively. The second inhibitor variable z operates on a slower timescale, which is expressed by the small parameter ε, that is, $0 < \varepsilon \ll 1$.

In order to understand the full set of possible dynamics, one needs to consider the three-dimensional phase space spanned by x, y, and z. The two-dimensional case can be recovered for very small ε, which is equilvalent to a slowly varying variable z. Then, this quantity corresponds to a parameter in the equations for x and y, which yields the same functional dependencies as in Eqs. (4.2), Section (4.1).

Further information about this three-dimensional version of the Hindmarsh-Rose model can also be found in Refs. [BAR11a, HIN84, HIN82]. In particular, Ref. [BAR11a] contains a series nicely visualized examples of the time series and various one- and two-dimensional parameter scans.

5. Integrate-and-Fire Model

So far we have discussed simple neural models that can be described by ordinary differential equations with 2 variables, sometimes called one activator and one inhibitor as in the FitzHugh-Nagumo system. The following chapter focuses on an even simpler model. As we will see, this integrate-and-fire model consists of one variable only and realizes the firing process via a discontinuous reset of its variable. In addition networks of many elements are easily taken into account by simple update rules.

Consider a network of N neurones. The integrate-and-fire model uses a single variable u_i for each neural element $i = 1, 2, ..., N$, where u_i can be interpreted as membrane potential. Below a threshold $\theta \in R$, the dynamics of the individual elements is governed by function $F : \mathbb{R} \to \mathbb{R}$:

$$\dot{u}_i = F(u_i), \qquad i = 1, \ldots, N. \tag{5.1}$$

The variable $u_i(t)$ is also called a growth function. In a very simple case, F is given by $F(u) = u$ and the initial condition is set to zero for convenience, $u(t = 0) = u_0 = 0$. Thus, the solution of the free-running system below the threshold is given by

$$u_i(t) = e^t - 1. \tag{5.2}$$

If the i-th element reaches the threshold θ, its membrane potential u_i is reset to a pre-defined value of $\rho < \theta$, that is,

$$u_i(t_-) = \theta \to u_i(t_+) = \rho. \tag{5.3}$$

Here, $u_i(t_-)$ and $u_i(t_+)$ denote the left or right sided limits of u_i at time t

$$\lim_{\epsilon \to 0}(t - \epsilon) = t_- \quad \text{and} \quad \lim_{\epsilon \to 0}(t + \epsilon) = t_+, \tag{5.4}$$

respectively. In order to implement coupling and interaction effects, assume that the i-th neuron fires at time t. If two neurons, i and j, are coupled to each other, they interact as follows

$$u_i(t_-) = \theta \Rightarrow u_j(t_+) = u_j(t_-) + \epsilon_{ij}. \tag{5.5}$$

In addition element i is reset to ρ as described above. This is depicted in Fig. 5.1. For the time being, we consider only undirected networks, where $\epsilon_{ij} = \epsilon_{ji}$ and we have $\epsilon_{ij} = 0$ if there is no connection between neurons i and j. The case

$$u_j(t_-) + \epsilon_{ij} \leq \theta, \tag{5.6}$$

it is known as subthreshold input. It means that the membrane potential $u_j(t_+)$ after the input is still below the threshold value. If, however, a unit j crosses the threshold due to a pulse from element i,

$$u_j(t_-) + \epsilon_{ij} \geq \theta, \tag{5.7}$$

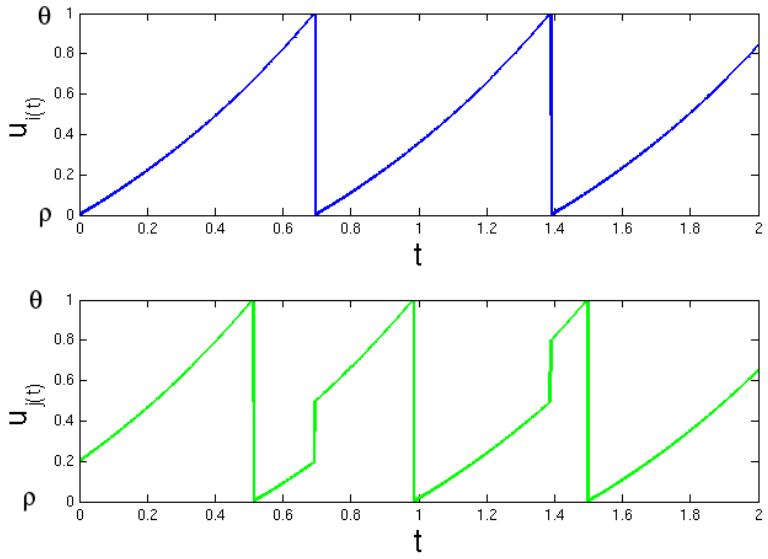

Fig. 5.1.: Sample traces of two coupled integrate-and-fire models with connectivity $\epsilon_{ij} > 0$. Top panel: At time $t = t^{(1)} \approx 0.7$, unit i reaches the threshold $\theta = 1$ and its membrane potential is reset to $\rho = 0$. It generates a pulse which is sent to the units j and k. Bottom panel: Unit j receives the pulse and its membrane potential is increased to $u_j\left(t_-^{(1)}\right) + \epsilon_{ji}$. The pulse is subthreshold, since $u_j\left(t_-^{(1)}\right) + \epsilon_{ji} < \theta$. Even the second pulse at $t^{(2)} = 1.4$ does not advance element j enough to induce a subsequent spike.

it is said to receive a suprathreshold input. As it reaches the threshold, it sends a pulse itself and is reset. This induced spiking yields an additional input to all neighbors of the element j. Subsequently, an avalanche might be triggered if the excitation of those neurons connected to element j in turn causes some of them to reach their threshold. This network-mediated excitation process ceases once no new oscillator reaches the threshold due to incoming signals.

There is a fine detail in the way the neural element is reset. Due to incoming signals from its neighbors, the value $u_j(t_-) + \epsilon_{ij}$ might be larger than the threshold θ, i.e., it experiences an ultrathreshold input. Now the response can be realized is either with or without partial reset.

In the model without partial reset it is not important how big is the ultrathreshold input, the membrane potential is always back on ρ. This can be also known as absorption. This could be interpreted as fast internal timescales compared to the signal transmission.

In the model with partial reset, the value after reset is described by the reset-

function R

$$u_j(t_-) + \epsilon_{ij} \geq \theta \Rightarrow u_j(t_+) = \rho + R\left(u_j - \theta\right). \tag{5.8}$$

Then the receiving element j keeps part of his surplus charge $u_j - \theta$. Consider, for instance, a linear reset-function

$$R_c(\xi) = c\xi, \tag{5.9}$$

where c is the reset parameter, $0 \geq c \geq 1$. The special case $c = 0$ recovers to the model without partial reset, and $c = 1$ means that the surplus charge, which is the amount of $u_j(t_-) + \epsilon_{ij}$ larger than θ, is completely maintained.

Finally, a short remark why this type of model is widely used. One of the reasons is that it makes simulations easier compared to the models of chapters 2, 3, and 4. By considering Eq. (5.2) for $u(t)$ one might have noticed that it is so easy to calculate t by inverting the equation. Moreover, by knowing the network structure ϵ_{ij} we are able to calculate all reset events, that is, the spike times, one after the other and the time that any element takes to reach the threshold. This means there is no need to numerically integrate the solutions. Instead we can analytically calculate the sequence of spikes and the time series of all elements.

Further information on this type of model can be found in Refs. [KIR08, HEE11a].

6. Izhikevich Neurons Model

6.1. Neuro-Computational Features

To understand how the spiking neurons of the brain work, we need to develop a model, which should have two requirements: 1) It should be computationally simple; and 2) It should be able to generate realistic firing patterns of neurons. For example, the Hodgkin-Huxley model, which we will briefly discuss in section 7.1, is precise and to large extent accurate in describing the physiology of a neuron, but it does not allow to simulate a large number of neurons due to its complexity. On the other hand, integrate-and-fire models are feasible for fast simulations of many neurons, but it is a very simple class of model compared to the physiological processes of real neurons.

In this chapter we first review the most important neuro-computational scenarios and then we introduce a simple model that claims to be plausible like the Hodgkin-Huxley model and at the same time also computationally efficient like an integrate-and-fire model.

A) Oscillatory regime: Most neurons get excited when they are stimulated by an input current. If the input current is continuously applied, neurons operate in an oscillatory regime as shown in Fig. 6.1(a).

B) One-time only excitation: Some certain neurons respond to a single action potential corresponding to the onset of an applied excitatory current input pulse and get excited at the onset of the input current and fire a spike. as in Fig. 6.1(b).

These two dynamical scenarios are already discussed in previous chapters. In addition to oscillatory and excitable behavior, there are also other modes of operation. Some of them will be described in the following.

C) Periodic bursting: Neurons have the ability to generate action potentials or spikes. Some neurons under certain conditions exhibit bursting behavior, in which long periods of quiescence are interrupted by a rapid firing of several spikes and a subsequent return to the quiescent state as depicted in Fig. 6.1(c). Bursting is primarily the result of a slow current input.

D) One-time bursting: Similar to the neurons which exhibit bursting, there are others that exhibit one-time busting. These neurons report the beginning of the simulation by transmitting a burst. See Fig. 6.1(d).

The nullclines corresponding to the cases (A) to (D) are shown in Fig. 6.2. Of course, there are many more neuro-computational scenarios for neurons. Details can be found, for instance, in Refs. [IZH03, IZH04].

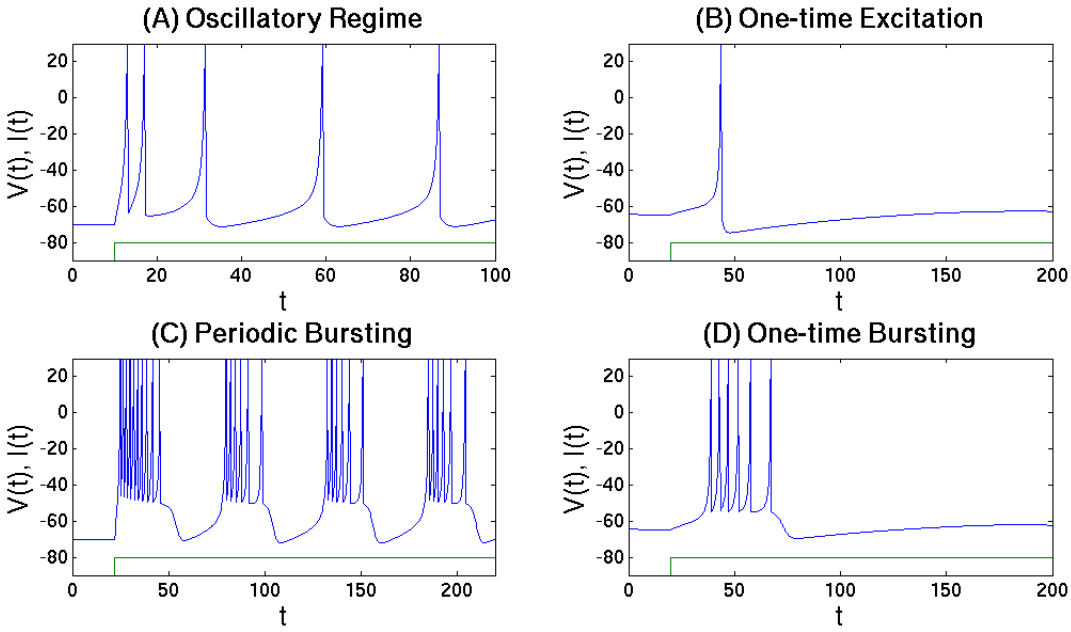

Fig. 6.1.: Summary of the neuro-computational properties of biological spiking neurons. Blue diagrams show the $V(t)$ and green diagrams show $I(t)$. for all panels $a = 0.02$, (A): $b = 0.2$, $c = -65$, $d = 6$ and $V = -70$, (B):$b = 0.25$, $c = -65$, $d = 6$ and $V = -64$, (C): $b = 0.2$, $c = -50$, $d = 2$ and $V = -70$, (D): $b = 0.25$, $c = -55$, $d = 0.05$ and $V = -64$.

6.2. Simple Model of Spiking Neurons

In this section, we present a simple spiking model that is as biologically plausible as the Hodgkin-Huxley model, yet as computationally efficient as the integrate-and-fire model [IZH03, IZH04]. All of the scenarios depicted in Fig. 6.1 are obtained using this a simple model of spiking neurons, which was proposed by Izhikevich. The model equations are given by:

$$\dot{v} = 0.04v^2 + 5v + 140 - u + I \tag{6.1a}$$

$$\dot{u} = a(bv - u) \tag{6.1b}$$

with system parameters a and b. I denotes a constant input. The variable v corresponds to the membrane potential of a neuron and u can be understood as a recovery variable. Physiologically, the latter accounts for the activation of potassium currents and inactivation of sodium currents and act as an inhibitory feedback to v. The parameter a addresses different timescales of v and u. The parameter b describes the sensitivity of the recovery variable u to the subthreshold fluctuations of the membrane potential v.

In addition the model uses a reset mechanism similar to integrate-and-fire models.

In detail, the variables u and v are reset at $v \geq 30$mV. Then the dynamical variable v is reset to c and u is advanced as $u + d$ with dimensionless parameters c and d. The first equation (6.1b) is motivated by bifurcation theory and normal-form reduction similar to previous chapters. Due to its functional dependence, $\dot{v} = v^2 + I$, the model is sometimes referred to as quadratic integrate-and-fire neuron. Depending on the choice of parameters a, b, c, and d this model can exhibit firing patterns of various types of cortical neurons.

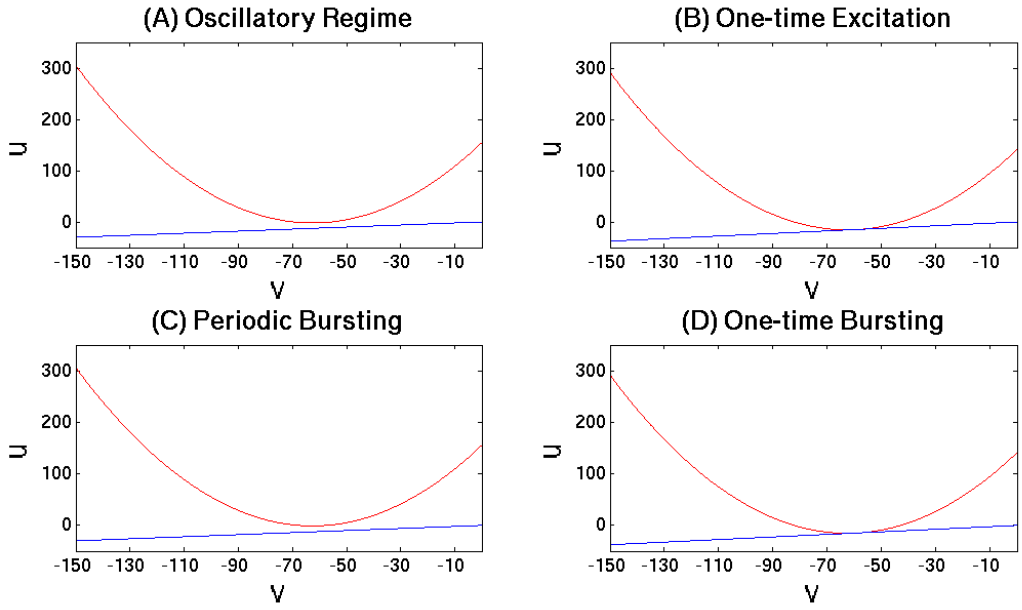

Fig. 6.2.: Nullclines of the dynamical scenarios discussed in Fig. 6.1.

Random Process: or stochastic process is a process used to introduce the evolution of random variables over time. Stochastic processing networks are used as models for complex processing systems involving dynamic interactions subject to uncertainty, in other words its a model or a process of generating data usually with power law structure to understand complex networks like brain. In the model we explained in this section Izhikevich used Poisson statistics for probability distribution.

In the cortex, the timing of successive action potentials appears to be highly irregular. Thus, we can approximatively assume that the irregular interspike interval reflects a random process. Furthermore, we assume that the generation of each spike depends only on an underlying continuous/analog driving signal that we will refer to as the instantaneous firing rate. It follows that the generation of each spike is independent of all the other spikes, hence we refer to this as the independent spike hypothesis. If the independent spike hypothesis were true, then the spike train would be completely described a particular kind of random process called a Poisson process.

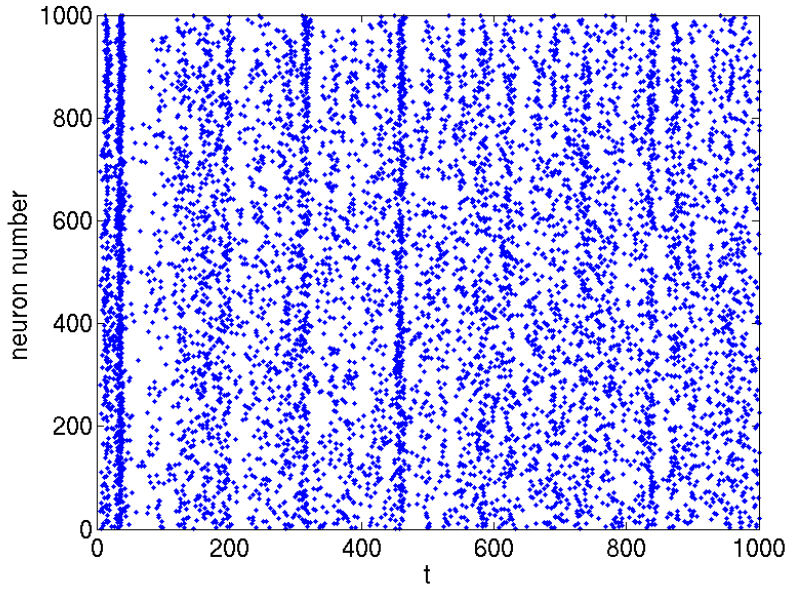

Fig. 6.3.: Simulation of a network with 1000 randomly connected, spiking neurons. Number of ecxitatory neurons = 800 and number of inhibitory neurons= 200, the initial value of V= -65.

Note that even though a Poisson spike train is generated by a random process, some stimuli could still evoke spikes very reliably by forcing the instantaneous firing rate to be very large at particular moments in time so that the probability of firing would then be arbitrarily close to 1.

For a more detailed information read Refs. [IZH03, IZH04].

7. More Physiological Models

The following chapter is a short overview of more physiological models. It is far from completion and not yet finished.

7.1. Hodgkin-Huxley Model

In this section we consider the Hodgkin and Huxley model, which carried out an analysis in the giant axon of the squid, with its half millimeter diameter (the typical axon in cortex has a diameter more than one thousand times smaller). In order to eliminate the complexity of the cable, a highly conductive axial wire was inserted inside the wire. This so called space clamp the potential along the entire axon spatially uniform, similar to the situation occurring in a patch of membrane. This, together with voltage clamping the membrane and the usage of pharmacological agents to block various currents, enabled Hodgkin and Huxley to dissect the membrane current into its constitutive components. The total membrane current is the sum of the ionic currents and the capacitive current. Using Kirchhoff's law and Fig. 7.1 to describe the axonal membrane, we have:

$$I_m(t) = I_{ionic}(t) + C_m \frac{dv(t)}{dt}. \tag{7.1a}$$

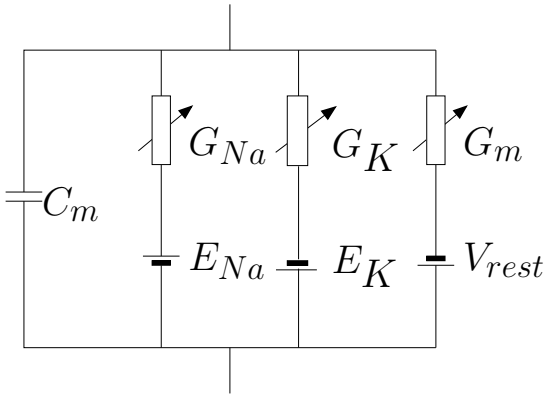

Fig. 7.1.: Electrical circuit for a patch of the squid axon modeled by Hodgkin and Huxley: the membrane of the squid axon is described by four parallel branches, that is, two passive branches with the membrane capacitance C_m and the leak conductance $G_m = 1/R_m$ and two time and voltage dependent branches representing the sodium and the potassium channels.

The action potential involves two major voltage dependent ionic conductances, a sodium conductance G_{Na} and a potassium conductance G_K. They are independent from each other. A third smaller so called leak conductance which we term G_L is

independent of the membrane potential. The total ionic current owing is the sum of sodium, potassium and the leak current:

$$I_{ionic} = I_{Na} + I_K + I_{leak}. \tag{7.2}$$

The individual ionic currents $I_i(t)$ are linearly related to the driving potential via Ohm's law:

$$I_i(t) = G_i(V(t))(V(t) - E_i). \tag{7.3}$$

Each of the two ionic conductances is expressed as a maximum conductance G_{Na} and G_K multiplied by a numerical coefficient representing the fraction of the maximum conductance actually open.Hodgkin and Huxley determined that the squid axon carries three major currents: voltage gated persistent K^+ current with four activation gates (resulting in term n^4 in the Eq. (7.1)); voltage gated transient Na^+ current with three activation gates and one inactivaion gate (the term m^3h in Eq. (7.1)), and leak current I_L which is carried mostly by Cl^- ions.

In the following we describe the dynamics of each considered currents. The potassium Current I_K can be written as

$$I_K = G_K n^4 (V - E_K) \tag{7.4}$$

and a dynamical equation for a so-called gating variable n

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V)n, \tag{7.5}$$

describes the state of a fictional activation particle, a dimensionless number between 0 and 1. α_n (resp. β_n) is a voltage-dependent rate constant (in units of $1/s$), specifying how many transitions occur between the closed and open state (resp. from the open to closed state):

$$\alpha_n(V) = 0.01\frac{10 - V}{e^{(10-V)/10} - 1} \tag{7.6a}$$

$$\beta_n(V) = 0.125e^{-V/80}. \tag{7.6b}$$

Similar to the potassium current, the sodium current I_{Na} is given by

$$I_{Na} = G_{Na} m^3 h (V - E_{Na}). \tag{7.7}$$

Here we have two gating variables m and h. The corresponding dynamical equations are similar to Eq. (7.5):

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V)m \tag{7.8a}$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V)h. \tag{7.8b}$$

Again m and h are dimensionless numbers with $m \geq 0$ and $h \leq 1$. And

$$\alpha_m(V) = 0.1\frac{25-V}{e^{(25-V)/10}-1} \tag{7.9a}$$

$$\beta_m(V) = 4e^{-V/18} \tag{7.9b}$$

$$\alpha_h(V) = 0.07e^{-V/20} \tag{7.9c}$$

$$\beta_h(V) = \frac{1}{e^{(30-V)/10}+1}. \tag{7.9d}$$

Instead of using rate constants α_n and β_n, we can use voltage-dependent time constants and steady-state values.

$$\tau_n = \frac{1}{(\alpha_n+\beta_n)} \tag{7.10a}$$

$$\tau_m = \frac{1}{(\alpha_m+\beta_m)} \tag{7.10b}$$

$$\tau_h = \frac{1}{(\alpha_h+\beta_h)}, \tag{7.10c}$$

the steady state activation funcions n_∞ and m_∞ give the asympotic value of n nad m, respctively, when the potenial is fixed. And h_∞ is the voltage sensitive steady sate inactivaion functions and they are calculated as follows:

$$n_\infty = \alpha_n/(\alpha_n+\beta_n) \tag{7.11a}$$

$$m_\infty = \alpha_m/(\alpha_m+\beta_m) \tag{7.11b}$$

$$h_\infty = \alpha_h/(\alpha_h+\beta_h). \tag{7.11c}$$

Figure 7.2 depicts the time constants(right panel) and steady state activation and inactivation (left panel) as a function of the relative membrane potential V for sodium activation m (green) and inactivation h (pink) and potassium activation n (black). The steady state sodium inactivation h is a monotonically decreasing function of V while the activation variables n and m increase with the membrane voltage.

Finally, we arrive at the complete model combining all ingredients discussed above:

$$C_m\frac{dV(t)}{dt} = I - G_K n^4(V-E_K) - G_{Na}m^3(V-E_{Na})G_L(V-E_L). \tag{7.12a}$$

In addition I denotes the applied current which we considered as $I = 0\mu A/cm^2$ that is injected via intracellular electrode. This nonlinear differential equation, together with the three, ordinary, linear, first-order differential equations specifying the evolution of the rate constants (as well as their voltage-dependencies) constitutes the four-dimensional Hodgkin and Huxley model for the space clamped axon or for a small patch of membrane.

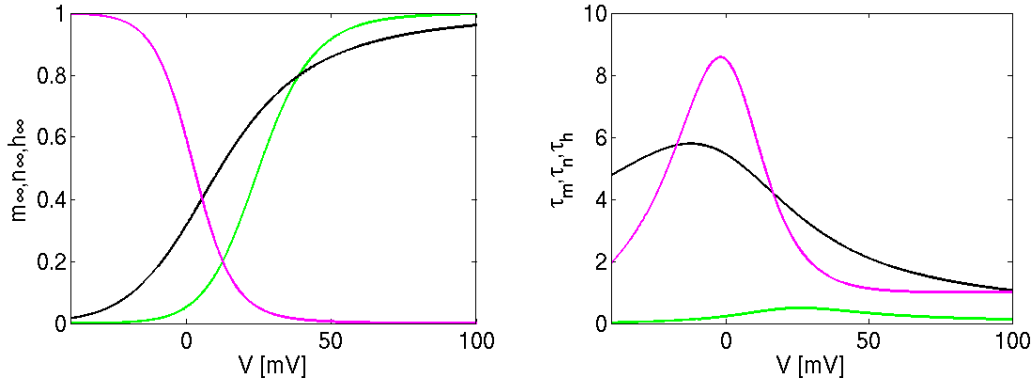

Fig. 7.2.: Left panel:Steady state activation and inactivaion functions. Right panel: Voltage dependent time constants, for sodium activation m (green) and inactivation h (pink) and potassium activation n (black).

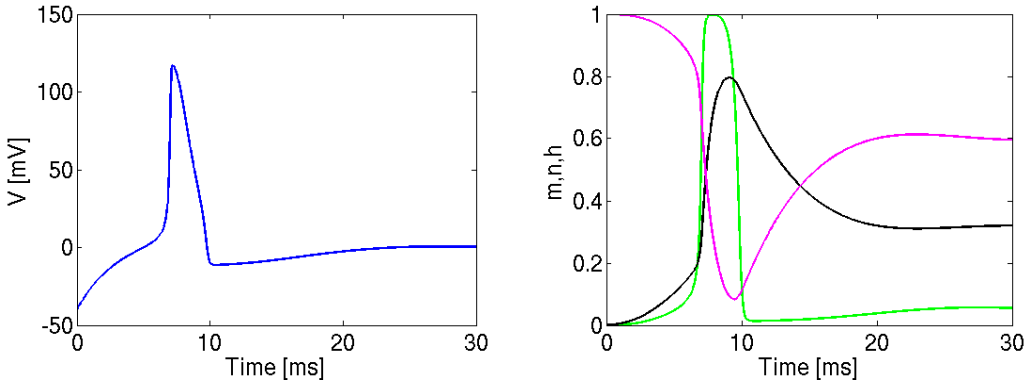

Fig. 7.3.: Left panel: Membrane voltage. Right panel: Activation potential, for sodium activation m (green) and inactivation h (pink) and potassium activation n (black).

Figure 7.3 displays the membrane potential in response to threshold and subthreshold stimuli. The injected current charges up the membrane capacity enabling sufficient I_{Na} to be recruited to outweigh the increase in I_K (due to the increase in driving potential). The smaller current pulse fails to trigger an action potential, but causes a depolarization followed by a small hyper-polarization due to activation of I_K.

In this section we used Izhikevich book **Dynamical Systems in Neuroscience** for the differenial equations and parameters, the diagrams and the curves generated are qualitatively similar to the curves of other versions like Koch book **Biophysics of Computation: Information Processing in Single Neurons**.

Further information can be found in Refs. [IZH07, KOC99].

7.2. Morris-Lecar Model

The dominant mode of communication between neurons happens via the generation and transmission of action potentials or spikes. Neurons accomplish this through the coordinated activity of voltage-sensitive ion channels in the cell membrane, which open and close in specific ways in response to changes in membrane voltage as seen in the previous chapter. The Morris-Lecar model is a two-dimensional simple model of this spike-generation process for a single neuron. It has the form [MOR81]:

$$C_m\dot{v} = I(t) - g_L(v - v_L) - g_{Ca}m_\infty(v)(v - v_{Ca}) - g_K\omega(v - v_K) \tag{7.13a}$$

$$\dot{\omega} = \phi(\omega_\infty(v) - \omega)/\tau_\omega(v). \tag{7.13b}$$

The variable v denotes the membrane voltage. The first equation (7.13a) expresses Kirchhoff's law with $I(t)$ representing a stimulus in the form of an incoming current. The variable ω is a gating variable, which describes the fraction of membrane ion channels. The original form of the model employed an instantaneously responding voltage-sensitive Ca_{2+} conductance for excitation and a delayed voltage-dependent K_+ conductance for recovery. In all physiocal details, neurons typically have multiple kinds of ion channels. Thus, in more realistic models like the Hodgkin-Huxley equations discussed in section 7.1, these are taken into account by different gating variables. The Morris-Lecar model is simplified in the sense that there is just one effective gating variable. The constant C_m is the membrane capacitance and ϕ a timescale parameter.

Spike generation depends crucially on the presence of voltage-sensitive ion channels that are permeable only to specific types of ions. The Morris-Lecar equations include just two ionic currents, i.e., calcium and potassium. The voltage response of ion channels is governed by the ω-equation (7.13b) and the auxiliary functions ω_∞, τ_ω, m_∞. The forms and meanings of the auxiliary functions and other parameters in Eqs. (7.13) are given in the following:

$$m_\infty(v) = \frac{1}{2}\left[1 + \tanh\left(\frac{v - v_1}{v_2}\right)\right] \tag{7.14a}$$

$$\omega_\infty(v) = \frac{1}{2}\left[1 + \tanh\left(\frac{v - v_3}{v_4}\right)\right] \tag{7.14b}$$

$$\tau_\omega(v) = \frac{1}{\cosh\left(\frac{v - v_3}{2v_4}\right)}. \tag{7.14c}$$

$m_\infty(v)$ and $\omega_\infty(v)$ are probability functions of conductance, which are derived from the assumption that, in equilibrium, the open and closed states of a channel are partitioned according to a Boltzmann distribution. In addition, one can see that the time constant τ_ω is voltage-dependent. We use the following values for simulations depicted below: $C_m = 20$, $g_L = 2$, $g_K = 8$, $g_{Ca} = 4$, $v_K = -84$, $v_L = -60$, and $v_{Ca} = 120$.

Let us consider a Morris-Lecar neuron with these parameters. In our simulations at $I_0 = 35$, the Morris-Lecar model is excitable because at large enough perturbations in voltage, an action potential is generated, while at small perturbations from rest, the voltage quickly decays back to the resting state. As shown in Figure 7.4 we can see a stable fixed point and a limit cycle in the phase portrait panel and a spike in the time series panel. On the other hand, at $I_0 = 39.5$, we find a continuous series of action potentials. Thus, we have a periodic solution to the Morris-Lecar equations for $I_0 = 39.5$, which is shown in Figure 7.5. The periodic orbit in phase space and the periodic bursting in the time series indicate the oscillatory regime in our dynamical system. We see qualitative changes in the dynamics of this system by changing the parameter I_0, which means that a bifurcation takes place in the system. This bifurcation observed in the Morris-Lecar model is a homoclinic bifurcation.

Further information can be found in Refs. [MOR81, LIN11a].

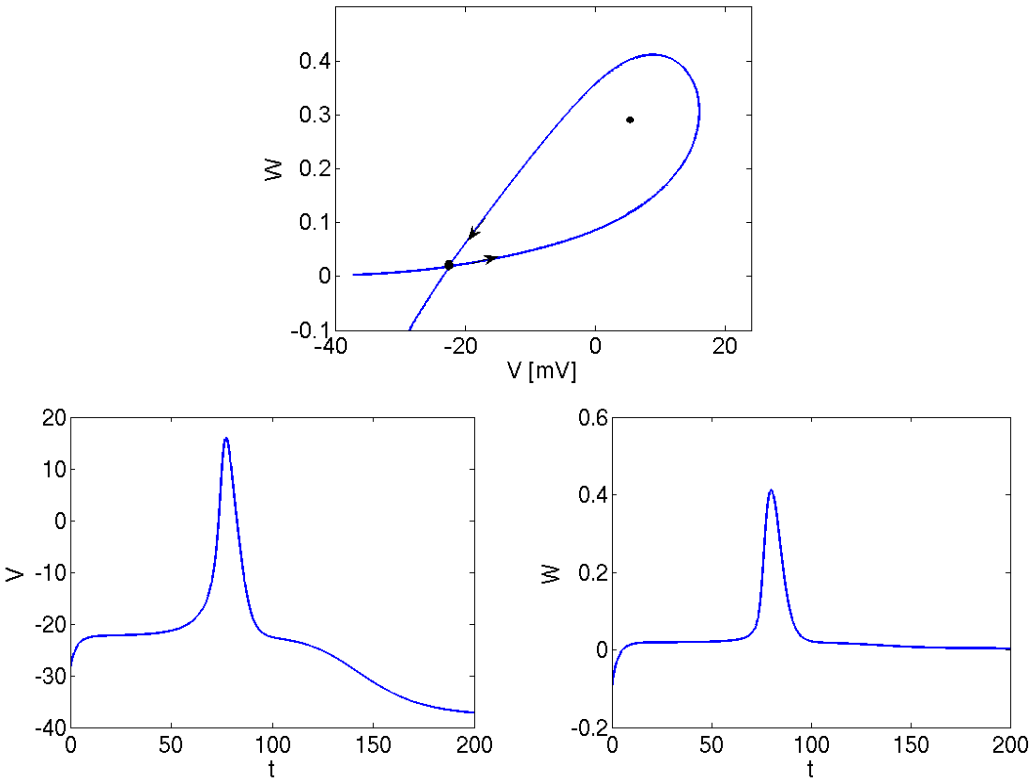

Fig. 7.4.: The phase portrait of the homoclinic bifurcation present in the Morris-Lecar model at $I_0 = 35$ and corresponding time series for v and ω.

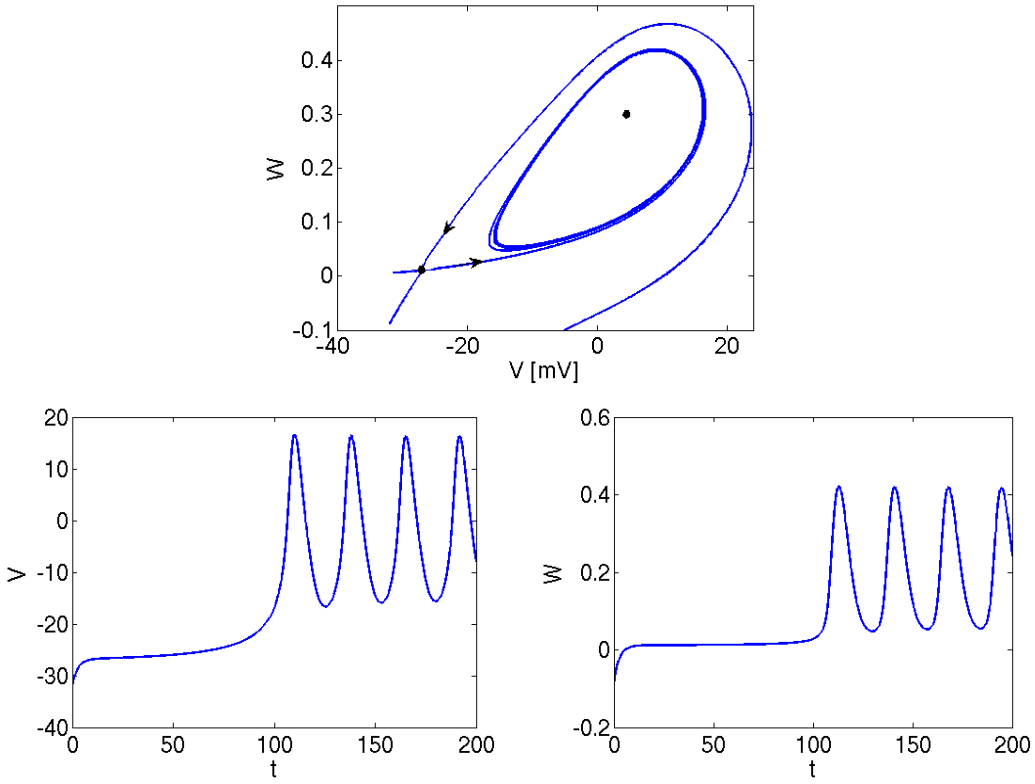

Fig. 7.5.: The phase portrait of Morris-Lecar at $I_0 = 39.5$ (oscillatory regime) and corresponding time series for v and ω.

References

[AUS07] R. Aust: *Multiple time-delayed feedback control of the coherence resonance in a model showing a global bifurcation*, Master's thesis, TU Berlin (2007).

[AUS09] R. Aust, P. Hövel, J. Hizanidis, and E. Schöll: *Delay control of coherence resonance in type-I excitable dynamics*, Eur. Phys. J. ST **187**, 77–85 (2010).

[BAR11a] R. Barrio and A. Shilnikov: *Parameter-sweeping techniques for temporal dynamics of neuronal systems: case study of hindmarsh-rose model*, J. Math. Neuroscience **1**, 1–22 (2011).

[BAR12b] A.-L. Barabási: *Network science*. ebook (2012).

[DIT94] T. Ditzinger, C. Z. Ning, and G. Hu: *Resonancelike responses of autonomous nonlinear systems to white noise*, Phys. Rev. E **50**, 3508 (1994).

[FIT61] R. FitzHugh: *Impulses and physiological states in theoretical models of nerve membrane*, Biophys. J. **1**, 445–466 (1961).

[HAU06] B. Hauschildt, N. B. Janson, A. G. Balanov, and E. Schöll: *Noise-induced cooperative dynamics and its control in coupled neuron models*, Phys. Rev. E **74**, 051906 (2006).

[HEE11a] A. Heesing: *Integrate-and-fire neurons with partial reset*. Bachelor thesis, Technische Universität Berlin (2011).

[HIN82] J. L. Hindmarsh and R. M. Rose: *A model of the nerve impulse using two first-order differential equations*, Nature **296**, 162 (1982).

[HIN84] J. L. Hindmarsh and R. M. Rose: *A model of neuronal bursting using three coupled first order differential equations*, Proc. Roy. Soc. B **221**, 87 (1984).

[HIZ07a] J. Hizanidis, R. Aust, and E. Schöll: *Delay-induced multistability in a generic model for excitable dynamics*, in *Proc. ASME 2007 Des. Eng. Techn. Conf. Comp. Inform. Eng. Conf.*, edited by T. Kalmár-Nagy (ASME, Las Vegas, 2007), pp. 1–8, DECT2007-34329.

[HIZ07] J. Hizanidis, R. Aust, and E. Schöll: *Delay-induced multistability near a global bifurcation*, Int. J. Bifur. Chaos **18**, 1759–1765 (2008).

[HOD52] A. L. Hodgkin and A. F. Huxley: *A quantitative description of membrane current and its application to conduction and excitation in nerve*, J. Physiol. **117**, 500 (1952).

[HOP97] F. C. Hoppensteadt and E. M. Izhikevich: *Weakly connected neural networks* (Springer, New York, 1997).

[HOE09] P. Hövel, M. A. Dahlem, and E. Schöll: *Control of synchronization in coupled neural systems by time-delayed feedback*, Int. J. Bifur. Chaos **20**, 813–815 (2010).

[HU93a] G. Hu, T. Ditzinger, C. Z. Ning, and H. Haken: *Stochastic resonance without external periodic force*, Phys. Rev. Lett. **71**, 807 (1993).

[IZH03] E. M. Izhikevich: *Simple model of spiking neurons*, IEEE Trans. on Neur. Netw. **14**, 1569–1572 (2003).

[IZH04] E. M. Izhikevich: *Which model to use for cortical spiking neurons?*, Neural Networks, IEEE Transactions on **15**, 1063–1070 (2004).

[IZH07] E. M. Izhikevich: *Dynamical Systems in Neuroscience* (MIT Press, Cambridge, MA, 2007).

[KIR08] C. Kirst and M. Timme: *From networks of unstable attractors to heteroclinic switching*, Phys. Rev. E **78**, 065201 (2008).

[KOC99] C. Koch: *Biophysics of Computation: Information Processing in Single Neurons* (Oxford University Press, New York, 1999).

[LIN11a] K. K. Lin, K. C. A. Wedgwood, S. Coombes, and L. S. Young: *Limitations of perturbative techniques in the analysis of rhythms and oscillations* (2011).

[MOR81] C. Morris and H. Lecar: *Voltage oscillations in the barnacle giant muscle fiber.*, Biophys. J. **35**, 193 (1981).

[NAG62] J. Nagumo, S. Arimoto, and S. Yoshizawa.: *An active pulse transmission line simulating nerve axon.*, Proc. IRE **50**, 2061–2070 (1962).

[NEW10] M. E. J. Newman: *Networks: an introduction* (Oxford University Press, Inc., New York, 2010).

[SCH08e] F. M. Schneider: *Beeinflussung von neuronalen Erregungswellen durch raum-zeitliche Rückkopplung*, Master's thesis, Technische Universität Berlin (2008).

[STA10] C. J. Stam: *Characterization of anatomical and functional connectivity in the brain: A complex networks perspective*, Int. J. Psychophysiol. **77**, 186–194 (2010), PROCEEDINGS OF THE 15TH WORLD CONGRESS OF PSYCHOPHYSIOLOGY of the International Organization of Psychophysiology (I.O.P.) Budapest, Hungary September 1-4, 2010.

[STR94a] S. H. Strogatz: *Nonlinear Dynamics and Chaos* (Westview Press, Cambridge, MA, 1994).

[TSU07] S. Tsuji, T. Ueta, H. Kawakami, H. Fujii, and K. Aihara: *Bifurcations in two-dimensional Hindmarsh-Rose type model*, Int. J. Bifur. Chaos **17**, 985 (2007).

A. A Brief Introduction to Networks

Behind any complex system – and one of the most complex known to us is the brain – lies an intricate wiring diagram or a network that defines the interactions between the components. Thus we never understand complex systems unless we are able to describe and understand the networks (graphs) behind them. The following appendix briefly summarizes the basic terminology and characteristic quantities descibing networks. It is far from completion and nor will it ever be. For a detailed introduction to networks see Refs. [NEW10, BAR12b].

A.1. Graph Theory

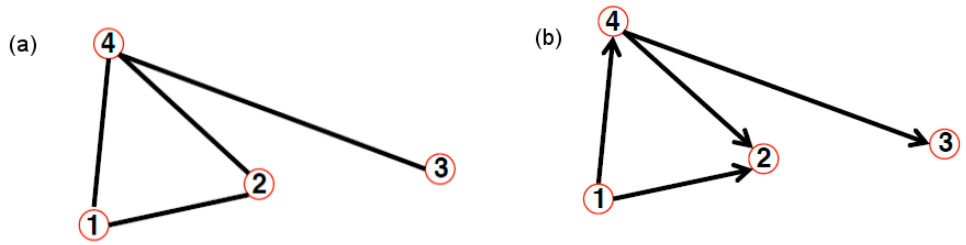

Fig. A.1.: (a) undirected and (b) directed networks.

Graph theory is the study of graphs, mathematical structures used to model pairwise relations between objects from a certain collection. A graph or a network in this context is a collection of vertices or nodes and a collection of edges that connect pairs of vertices. The network may be undirected as in Fig. A.1(a) or directed as in Fig. A.1(b). More formally, we define a network G as an ordered pair $G = (N, L)$, where N is a set of nodes (vertices) V, and L is a set of edges (links) E.

A.2. Various Characteristics

Adjacency matrix: This is an $n \times n$ matrix $\mathbf{A}$, where n is the number of vertices in the network. If there is an edge from a vertey x j to a vertex i, then the element a_{ij} is 1, otherwise it is 0. As an example, the adjacency martix of the undirected in Fig. A.1(a)is:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \tag{A.1}$$

Node degree k_i: the number of links connected to the node i.

Avarage degree $\langle k \rangle$: a measure of how many edges are in set L compared to number of vertices in set N

$$\langle k \rangle = \frac{1}{N} \Sigma k_i, \tag{A.2}$$

where N is the number of nodes.

Degree distribution $P(k)$: the probability that a randomly chosen node has degree k. In a random network with connection probability p the degree k_i of a node i follows a binominal distribution with parameters $N-1$ and p:

$$P(k_i = k) = C_{N-1}^{k} p^k (1-p)^{N-1-k}, \tag{A.3}$$

or a Poisson distribution in the limit of large N. Real-world networks like World Wide Web, Internet or social networks are mostly found to be very different from the random network in their degree distributions. They almost always follows a power law

$$P(k) \sim k^{-\gamma}, \tag{A.4}$$

where γ is a constant.

Diameter of a network: the maximal distance between any pair of its nodes. Random networks tend to have small diameters.

Average path length: the average distance between any pair of nodes in a network, and is a way to characterize the spread of a network. It is notable that the average path length of a real network is close to the average path length of a random network.

$$\ell \sim \frac{\ln(N)}{\ln(\langle k \rangle)}. \tag{A.5}$$

Clustering coefficient: If we consider a node in a random network and its nearest neighbors, the probability that these neighbors are also connected to one another is described with clustering coefficient.

Local clustering coefficient: a measure introduced by Watts and Strogatz and defined as follows:

$$C_i = \frac{2e_i}{k_i(k_i-1)}, \tag{A.6}$$

where e_i is the number of neighbors of a specific node which are also connected to one another.

Global clustering coefficient: also known as transitivity

$$T = \frac{3 * \text{number of triangels in G}}{\text{number of connencted triples of vertices in G}}. \tag{A.7}$$

A.3. Regular networks

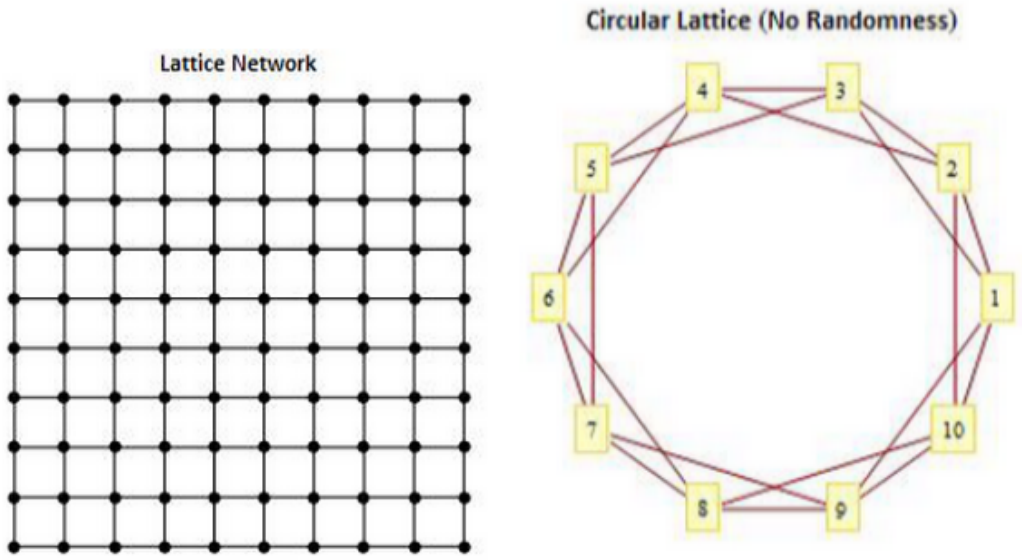

Fig. A.2.: Regular networks.

In regular networks each node has exactly the same number of links, and the network looks the same from every single node. Regular networks are highly ordered and highly clustered. A regular network architecture, such as the lattice, does not offer high connectivity in many cases because of its large average path length.In other words a long and circuitous route is required to reach many nodes.

A.4. Random networks

A random network, is a theoretical construct which contains links that are chosen completely at random with equal probability. A randoim network is considered to be highly disordered. Using a random number generator, one assigns links from one node to a second node. In a random network, unlike the regular architecture, the topological rule is that the node degrees may not all be equal. Instead, the degrees are distributed according to a Poisson distribution because it is assumed that any linking between nodes can happen with equal probability. Random links typically result in shortcuts to remote nodes, thus they have small average path length. Unlike real world networks, there is low clustering in random networks. Therefore, the resulting

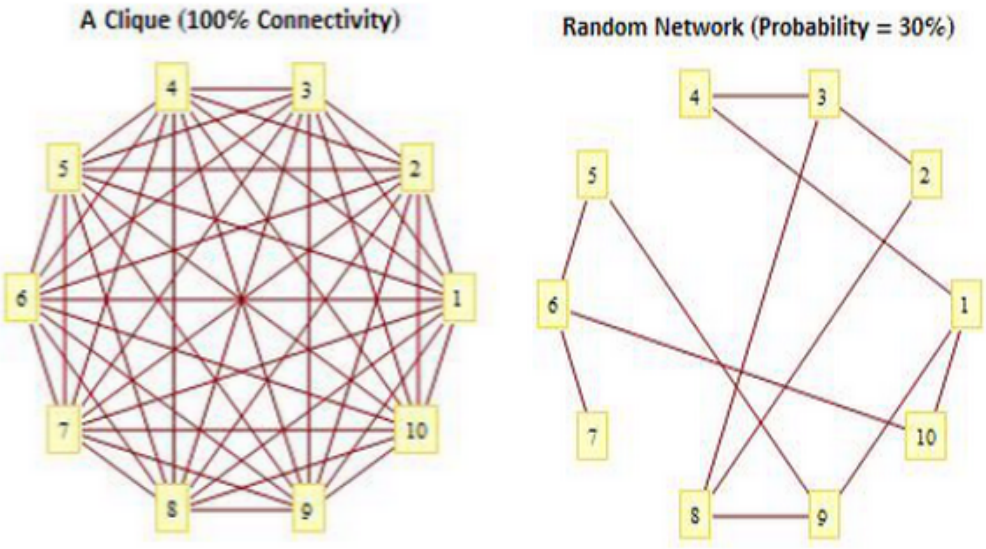

Fig. A.3.: Random networks.

network very rarely contains highly connected nodes. Consequently, a random network is not a good candidate model for the highly connected architecture that characterize real world networks.

A.5. Watts and Strogatz model, Small World networks

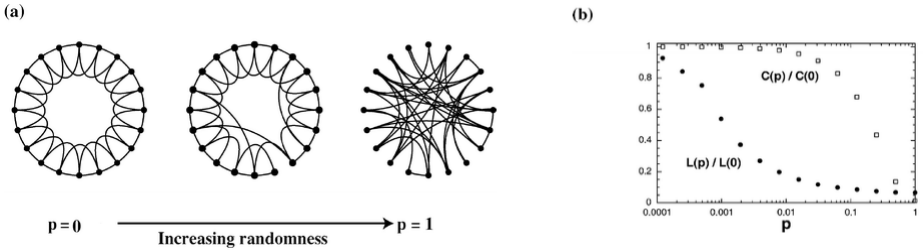

Fig. A.4.: Watts and Strogatz model.

Watts and Strogatz proposed a model between a regular lattice and a random network. They start with aring lattice and randomly rewired each edge of the lattice with probability p, such that self-connections and duplicate edges are excluded. By varying p one can closely monitor the transition between order ($p = 0$) and randomness ($p = 1$) as shown in Fig. A.4(a)
To understand the coexistence of small path length and clustering, we study the behavior of the clustering coefficient $C(p)$ and the avarage shortest path length $\ell(p)$ as a function of the rewiring probability p and the diagram gained by Watts and

Strogatz is shown in Fig. A.4(b). This regime originates in a rapid drop of $\ell(p)$ for small values of p, while $C(p)$ stays almost unchanged, resulting in networks that are clustered but have a small avarage shortest path length, which is in a excellent agreement with characteristics of real networks. The simplest organism studied is C. elegans nervous system which consists of 302 neurons and its pattern of connectivity has been completely mapped. This fact motivated Watts and Strogatz to examine their idea about small-world network on it. They found out that the central nervous system of C. elegans has a relatively high clustering and a short path length, so it could be considered as an example for small-world network.

A.6. Brain as a complex network

Human brain is estimated to have about 10^{10} to 10^{11} neurons, it is a very large network such as Internet and the World Wide Web. MRI DTI technology made it possible to evaluate neural pathways within the brain, brainstem, or spine, such as motor-skill controls and speech. In a study demonstrated by Li et al. (2009) it is displayed that brain networks have high clustering and short path length typical of small-world networks. They also found that a higher IQ is strongly correlated with a larger number of connections and a shorter path length and a higher efficiency. Without connections the brain is useless, so considering brain as a complex network, the main question is: How are the neurons connected? There is only an average of 10^4 connections per neuron in the brain network which characterizes a complex network with one of the lowest connectivity levels. There is a biological ?price? for each connection in terms of biological material and energy; therefore a brain that achieves the same information processing at a lower wiring cost is likely to be at a significant evolutionary advantage (Ref. [STA10]).

B. Numerical Methods to Solve Differential Equations

Numerical methods are commonly used for solving mathematical problems that are formulated in science and engineering where it is difficult or even impossible to obtain exact solutions. Only a limited number of differential equations can be solved analytically. Numerical methods, on the other hand, can give an approximate solution to (almost) any equation. An ordinary differential equation (ODE) is an equation that contains an independent variable, a dependent variable, and derivatives of the dependent variable. Literal implementation of this procedure results in Euler Method.

B.1. Euler Method

In mathematics and computational science, the Euler method, named after Leonhard Euler, is a numerical procedure for solving ordinary differential equations (ODEs) with given initial values. Consider the problem of approximating a continuous function $y = f(x)$ on $x \geq 0$ which satisfies the differential equation:

$$\frac{dy}{dx} = f(x, y). \tag{B.1}$$

on $x > 0$ and the initial condition $y(0) = \alpha$, in which α is a given constant. The basic idea in Euler Method is the definition of a derivative:

$$\dot{y}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{B.2}$$

For small amounts of h $(h > 0)$, we can rewrite the equation as:

$$f(x+h) \approx f(x) + hf(x, y), \tag{B.3}$$

which yields the approximation

$$y(x+h) \approx y(x) + hf(x, y(x)). \tag{B.4}$$

This enables one to approximate $y(x+h)$ in terms of $y(x)$ and $f(x, y(x))$. Equation (B.4) is the cornerstone of Euler Method. Explicit and implicit methods are approaches used in numerical analysis for obtaining numerical solutions of time-dependent ordinary differential equations, as is required in computer simulations of physical processes. Explicit methods calculate the state of a system at a later time from the state of the system at the current time, while implicit methods find a solution by solving an equation involving both the current state of the system and the later one.

Explicit Euler The general algorithm of the explicit Euler method, also known as forward Euler algorithm, considering the following definitions:

$$\frac{dx}{dy} = f(x, y); x(0) = A \tag{B.5a}$$

$$\frac{dy}{dx} = g(x, y); y(0) = B \tag{B.5b}$$

is based on the following approximation:

$$\frac{x^{k+1} - x^k}{\Delta t} = f(x^k, y^k) \tag{B.6a}$$

$$\frac{y^{k+1} - y^k}{\Delta t} = g(x^k, y^k), \tag{B.6b}$$

which can be rewritten as:

$$x^{k+1} = \Delta t f(x^k, y^k) + x^k \tag{B.7a}$$

$$y^{k+1} = \Delta t g(x^k, y^k) + y^k. \tag{B.7b}$$

Having the initial values of $x(0)$ and $y(0)$ in the above equations, the values for x^{k+1} and y^{k+1} can easily be calculated.

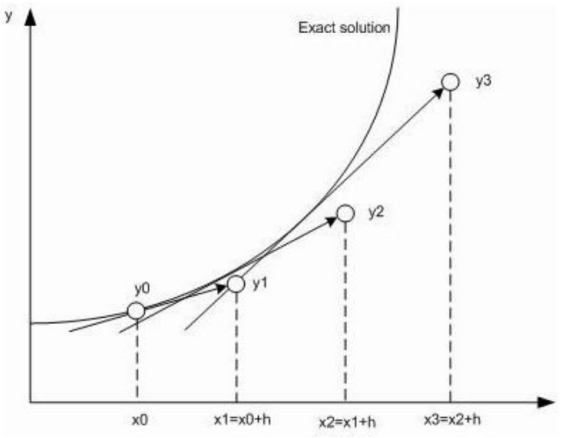

Fig. B.5.: Graphical illustration of the Explicit Euler method for an exponential like curve. Starting at point 1, the tangent of the curve is taken and linearly extrapolated to obtain point 2. There again the same procedure is used to obtain point 3. Note that point 2 lies on curve 2 and point three lies on curve 3, both of which are offset against the original curve.

Implicit Euler The implicit Euler method, which is also called backward Euler algorithm, is based on the backward difference approximation and written as:

$$\frac{x^{k+1} - x^k}{\Delta t} = f(x^{k+1}, y^{k+1}) \tag{B.8a}$$

$$\frac{y^{k+1} - y^k}{\Delta t} = g(x^{k+1}, y^{k+1}), \tag{B.8b}$$

which yields:

$$x^{k+1} = \Delta t f(x^{k+1}, y^{k+1}) + x^k \tag{B.9a}$$

$$y^{k+1} = \Delta t g(x^{k+1}, y^{k+1}) + y^k. \tag{B.9b}$$

It is clear that implicit method requires an extra computation (solving the above equation), and it can be much harder to implement. Implicit method is used in the case that the problem is stiff, for which the use of an explicit method requires impractically small time steps Δt to keep the error in the result bounded. For such problems, to achieve the given accuracy, it takes much less computational time to use an implicit method with larger time steps. Since Eq. (B.9) may be nonlinear, solving them in general requires an iterative solution method, such as functional iteration or Newton Raphson Method.

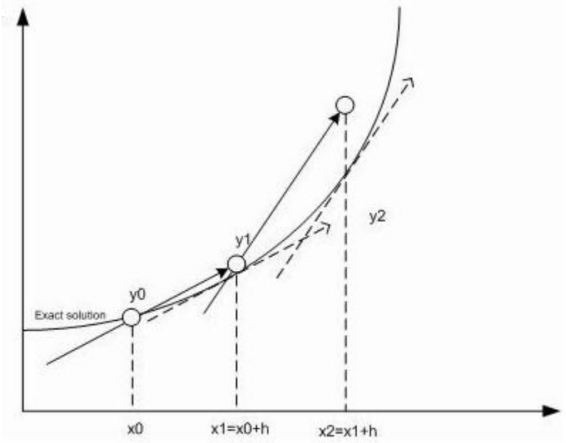

Fig. B.6.: Graphical illustration of the Implicit Euler Method. To obtain point 2 from point 1, we take the derivative at point 2 and extrapolate it at point 1. To obtain point 3 starting at point 2, we do the same: take the derivative at point 3 and extrapolate it at point 2.

Newton-Raphson Method The Newton-Raphson method is a relatively simple iteration for finding where a curve has a value of zero. This root finding algorithm is

a handy way of calculating difficult formulas, such as the square root of a number. The idea is to temporarily replace the complex curve with a simple straight line. The straight line, which approximates the curve but is far easier to work with, takes you closer to the answer and allows you to generate a new, better, straight line approximation. After only a few iterations you have an answer that is precise to a few decimal places. An important property of the Newton-Raphson method is that it works with no knowledge of the overall curve. It needs to know only the shape of the curve locally, at the current approximate answer. The Newton-Raphson method, or Newton Method, is a powerful technique for solving equations numerically. Like so much of the differential calculus, it is based on the simple idea of linear approximation. An initial guess for the root of the equation $f(x) = 0$ should be estimated, which is called x^0. From x^0, a (probably) improved estimate x^1 will be produced. From x^1, a new estimate x^2 will be produced and so on. This procedure goes on until x^k gets close enough to the root of the equation. Sometimes the initial estimation is called a guess. The Newton method is usually very good if x^0 is close to the root, and can be horrid if it is not. Therefore, the guess x^0 should be chosen with care. In the present case (to solve nonlinear equations in implicit Euler), x^k and y^k are considered as the initial values (guess) to calculate x^{k+1} and y^{k+1}. Considering a function H that has a zero at x^*, (i.e. $H(x^*) = 0$). By setting $y = H(x)$, changes in y and Δy could be approximated, due to changes in x and Δx, by linearizing H around some point x^i:

$$\Delta y = J_{H(x^i)} \Delta x, \tag{B.10}$$

where $J_{H(x^i)}$ is the Jacobian of H at x^i. In order to define the zero of H, Δy is assumed to be $-H(x^i)$ and then Eq. (B.9) should be solved for Δx to obtain:

$$\Delta x = -[J_{H(x^i)}]^{-1} H(x^i). \tag{B.11}$$

Because Eq. (B.10) is a Jacobian linearization of H, Eq. (B.11) provides a change in x that moves closer to the desired zero. A second iteration of this process can now be performed at a point x^{i+1} such that $x^{i+1} = x^i + \Delta x$. Using Eq. (B.11), this becomes:

$$x^{i+1} = x^i - [J_{H(x^i)}]^{-1} H(x^i). \tag{B.12}$$

Equation (B.12) can be used iteratively from some initial guess to yield a better approximation of x^*. The algorithm terminates once a value of x is reached such that $H(x)$ is sufficiently close to zero. In other words, every time the error should be calculated and compared to the acceptable error tolerance. Figure B.7 shows a graphical representation of iterations of the method.

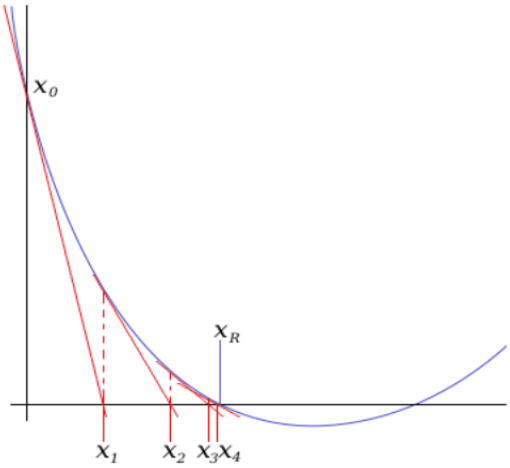

Fig. B.7.: The graphical illustration of the Newton-Raphson method.

C. MATLAB Source Code

C.1. Nonlinear models

```
function Projekt(S)
clear;
clc;
S = 2;

%% Given Parameter
b = 1; %0.7 or 1.0 or 1.05
beta = 0.67; %0.97 or 1.01
e = 0.1; %0.005
gamma = 0.5;

% Time span
t0 = 0;
tf = 10;
if S == 1
    dt = 0.002;
else
    dt = 0.001;
end
xt =t0:5:tf;

t = t0:dt:tf;
N = length(t);

%% Main Program
switch S
    case 1
        %==========SNIPER==========
        % Initial Values
        % 1,2,3 -> g,b,r (b=0.7)
        % 4,5,6 -> g,b,r (b=1.0)
        % 7,8,9 -> g,b,r (b=1.05)
        if b == 0.7
            x(1,:) = [1.3,0.98,-0.0001];
            y(1,:) = [1.5,1.5,-0.0001];
        elseif b == 1
            x(1,:) = [1.8,-1.5,0.0001];
            y(1,:) = [0.9,-0.8,0.0001];
        else
```

```
40              x(1,:) = [1.3,-1.5,0.0001];
41              y(1,:) = [1.5,-0.8,0.0001];
42          end
43
44          x_dot(1,:) = x(1,:) .* (1 - x(1,:).^2 - y(1,:)
               .^2) + y(1,:) .* (x(1,:) - b);
45          y_dot(1,:) = y(1,:) .* (1 - x(1,:).^2 - y(1,:)
               .^2) - x(1,:) .* (x(1,:) - b);
46
47          for i = 1:N - 1
48              x(i+1,:) = x(i,:) + dt * x_dot(i,:);
49              y(i+1,:) = y(i,:) + dt * y_dot(i,:);
50              x_dot(i+1,:) = x(i+1,:) .* (1 - x(i+1,:).^2
                    - y(i+1,:).^2) + y(i+1,:) .* (x(i+1,:)
                   - b);
51              y_dot(i+1,:) = y(i+1,:) .* (1 - x(i+1,:).^2
                    - y(i+1,:).^2) - x(i+1,:) .* (x(i+1,:)
                   - b);
52
53          end
54          Cx = sum(x(N,:))/3;
55          Cy = sum(y(N,:))/3;
56
57          % Diagrams of the results
58          figure(1);
59          if b ~= 1.05
60              p = plot(x(:,1),y(:,1),'g',x(:,2),y(:,2),'b
                   ',x(:,3),y(:,3),'r',Cx,Cy,'ok',0,0,'ok')
                   ;
61          else
62              p = plot(x(:,1),y(:,1),'g',x(:,2),y(:,2),'b
                   ',x(:,3),y(:,3),'r',0,0,'ok');
63          end
64          set(p,'LineWidth',2,'Markersize',10);
65          xlim([-1.5 1.5]);
66          ylim([-1.5 1.5]);
67          xlabel('$x$','interpreter','latex','FontSize'
               ,16);
68          ylabel('$y$','interpreter','latex','FontSize'
               ,16);
69          if b == 1.05
70              hold on
```

```
71              plot(-1.5:0.005:1.5,0,'-k',0,-1.5:0.005:
                    1.5,'-k');
72              hold off
73          else
74              hold on
75              plot(-1.5:0.005:1.5,0,'-k',0,-1.5:0.005:
                    1.5,'-k',Cx,-1.5:0.05:1.5,'k');
76              hold off
77          end
78
79          figure(2);
80          p = plot(t,x(:,1),'g',t,x(:,2),'b',t,x(:,3),'r'
                );
81          ylim([-1.5 1.5]);
82          set(p,'LineWidth',2);
83          set(gca,'XTick',xt);
84          xlabel('$time$','interpreter','latex','FontSize
                ',16);
85          ylabel('$x$','interpreter','latex','FontSize'
                ,16);
86          if b ~= 1.05
87              hold on
88              plot(t0:0.2:tf,Cx,'-k');
89              hold off
90          end
91
92          figure(3);
93          p = plot(t,y(:,1),'g',t,y(:,2),'b',t,y(:,3),'r'
                );
94          ylim([-1.5 1.5]);
95          set(p,'LineWidth',2);
96          set(gca,'XTick',xt);
97          xlabel('$time$','interpreter','latex','FontSize
                ',16);
98          ylabel('$y$','interpreter','latex','FontSize'
                ,16);
99          if b ~= 1.05
100             hold on
101             plot(t0:0.2:tf,Cy,'-k');
102             hold off
103         end
104
105         figure(4);
```

```
106         p = plot(x(:,1),x_dot(:,1),'g',x(:,2),x_dot(:
                ,2),'b',x(:,3),x_dot(:,3),'r');
107         set(p,'LineWidth',2);
108         xlabel('$x$','interpreter','latex','FontSize'
                ,16);
109         ylabel('$\dot{x}$','interpreter','latex','
                FontSize',16);
110
111         figure(5);
112         p = plot(y(:,1),y_dot(:,1),'g',y(:,2),y_dot(:
                ,2),'b',y(:,3),y_dot(:,3),'r');
113         set(p,'LineWidth',2);
114         xlabel('$y$','interpreter','latex','FontSize'
                ,16);
115         ylabel('$\dot{y}$','interpreter','latex','
                FontSize',16);
116
117     case 2
118
119         %==========Fitzhugh-Nagumo==========
120         % Initial Values
121
122         %          Canard     Subtresh     Supratresh
123         x(1,:) = [1             -1             0];
124         y(1,:) = [2/3           -0.75          0];
125
126         for j = 1:3
127             for i = 1:N-1
128                 if j==1
129                     x(i+1,j) = x(i,j) - (dt/e)*(x(i,j)
                            - x(i,j)^3/3 - y(i,j));
130                     y(i+1,j) = y(i,j) - dt*(x(i,j) -
                            gamma*y(i,j)+ beta);
131                 else
132                     G_1 = x(i,j);
133                     G_2 = y(i,j);
134                     G = NR(G_1,G_2);
135                     x(i+1,j) = G(1);
136                     y(i+1,j) = G(2);
137                 end
138             end
139         end
140
```

```
141         %x_dot & y_dot equals to zero
142         X = -5:0.001:5;
143         Y1= X - X.^3/3;
144         Y2 = (X + beta) / gamma;
145
146         % Diagrams of the results
147         figure(1);
148         p = plot(x(:,1),y(:,1),'m');
149         set(p,'LineWidth',1.5);
150         hold on
151         p = plot(x(:,3),y(:,3),'k');
152         set(p,'LineWidth',1.35);
153         hold on
154         p = plot(x(:,2),y(:,2),'g');
155         set(p,'LineWidth',1.35);
156         p = plot(X,Y1,'r',X,Y2,'b');
157         hold off
158         set(p,'LineWidth',1.5);
159         xlim([-2.5 2.5]);
160         ylim([-1.5 1.5]);
161         xlabel('u','FontSize',24);
162         ylabel('v','FontSize',24);
163         set(gca,'FontSize',16);
164
165          figure(2);
166          p = plot(t,x,'k');
167          ylim([-2.5 2.5]);
168          set(p,'LineWidth',2);
169          set(gca,'XTick',xt);
170          xlabel('$time$','interpreter','latex','
                FontSize',16);
171          ylabel('$x$','interpreter','latex','FontSize'
                ,16);
172
173          figure(3);
174          p = plot(t,y,'k');
175          ylim([-1.5 1.5]);
176          set(p,'LineWidth',2);
177          set(gca,'XTick',xt);
178          xlabel('$time$','interpreter','latex','
                FontSize',16);
179          ylabel('$y$','interpreter','latex','FontSize'
                ,16);
```

```
180
181             figure(4);
182             p = plot(x,x_dot,'k');
183             set(p,'LineWidth',2);
184             xlim([-2.5 2.5]);
185             xlabel('$x$','interpreter','latex','FontSize'
                    ,16);
186             ylabel('$\dot{x}$','interpreter','latex','
                    FontSize',16);
187
188             figure(5);
189             p = plot(y,y_dot,'k');
190             xlim([-1.5 1.5]);
191             set(p,'LineWidth',2);
192             xlabel('$y$','interpreter','latex','FontSize'
                    ,16);
193             ylabel('$\dot{y}$','interpreter','latex','
                    FontSize',16);
194
195 end
196
197 %% Newton Raphson Method
198 function F = NR(x,y)
199 tol = 1e-4;
200 H = [x;y];
201 eps = inf;
202 while eps > tol
203     F = [-H(1) + x + (dt/e)*(H(1) - H(1)^3/3 - H(2));
204          -H(2) + y + dt*(H(1) - gamma*H(2)+ beta)];
205     J(1,1) = -1 + (dt/e)*(1 - H(1)^2);
206     J(1,2) = -(dt/e);
207     J(2,1) = dt;
208     J(2,2) = -1 - (gamma*dt);
209
210     H = H - inv(J) * F;
211     eps = max(max(abs(F)));
212 end
213 F = H;
214 end
215 end
```

C.2. Hindmarsh-Rose model

```
1  %% Hindmarsh Rose
2  function HR2(S)
3  clear;
4  clc;
5  % S should be a ~ f
6  S = 'a';
7
8  %% Given Parameter
9  z = 0;
10 b = 1;
11 c = 3;
12
13 % Time span
14 t0 = 0;
15 tf = 20;
16 dt = 0.01;
17 t = t0:dt:tf;
18 N = length(t);
19
20 switch S
21     case 'a'
22         % Initial conditions
23         x(1,:) = [2  -2  -2.9  1.00  0.1  -2.9];
24         y(1,:) = [2  -1  -1.0  -1.0  0.1  1.50];
25
26         % Given parameters
27         a = 0.6;
28         d = 2;
29
30         % Calculation of the stable node
31         X_St = fsolve(@F,-3,optimset('Display','off'),a
               ,b,d);
32         Y_St = y_val(X_St);
33
34         % Calculation of the eigenvalues for the stable
               node
35         Lambda_St = R(1,-trace(X_St,b,c),deter(X_St,b,c
               ,d));
36
37         x_dot(1,:) = c * (x(1,:) - x(1,:).^3/3 - y(1,:)
               + z);
```

```
38          y_dot(1,:) = (x(1,:).^2 + d*x(1,:) - b*y(1,:) +
                a) / c;
39
40          for i = 1:N - 1
41              x(i+1,:) = x(i,:) + dt * x_dot(i,:);
42              y(i+1,:) = y(i,:) + dt * y_dot(i,:);
43              x_dot(i+1,:) = c * (x(i+1,:) - x(i+1,:)
                    .^3/3 - y(i+1,:) + z);
44              y_dot(i+1,:) = (x(i+1,:).^2 + d*x(i+1,:) -
                    b*y(i+1,:) + a) / c;
45          end
46
47          % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
48          X = -3:0.05:3;
49          Y1 = X - X.^3/3 + z;
50          Y2 = (X.^2 + d*X + a) / b;
51
52          % Diagram of the results
53          figure(1);
54          p1 = plot(x,y,'k',X,Y1,'.b',X,Y2,'.r');
55          set(p1,'LineWidth',2);
56          xlim([-3 3]);
57          ylim([-3 3]);
58
59          figure(2);
60          plot(t,x,'r',t,y,'b');
61
62      case 'b'
63          % Initial conditions
64          x(1,:) = [2.0  -2.0  -2.0  -1.00  -1.00  -2.9
                0.010  0.01];
65          y(1,:) = [2.0  -1.0  -0.5  -0.66  -0.68  1.50
                -0.01  0.01];
66
67          % Given parameters
68          a = 0.05;
69          d = 1.7;
70
71          % Calculation of the stable node
72          X_St = fsolve(@F,-3,optimset('Display','off'),a
                ,b,d);
73          Y_St = y_val(X_St);
```

```
74          % Calculation of the eigenvalues for the stable
                node
75          Lambda_St = R(1,-trace(X_St,b,c),deter(X_St,b,c
                ,d));
76
77          % Calculation of the unstable node
78          X_USt = fsolve(@F,0,optimset('Display','off'),a
                ,b,d);
79          Y_USt = y_val(X_USt);
80          % Calculation of the eigenvalues for the
                unstable node
81          Lambda_USt = R(1,-trace(X_USt,b,c),deter(X_USt,
                b,c,d));
82
83          % Calculation of the saddle node
84          X_Sa = fsolve(@F,-1,optimset('Display','off'),a
                ,b,d);
85          Y_Sa = y_val(X_Sa);
86          % Calculation of the eigenvalues for the saddle
                node
87          Lambda_Sa = R(1,-trace(X_Sa,b,c),deter(X_Sa,b,c
                ,d));
88
89          x_dot(1,:) = c * (x(1,:) - x(1,:).^3/3 - y(1,:)
                + z);
90          y_dot(1,:) = (x(1,:).^2 + d*x(1,:) - b*y(1,:) +
                a) / c;
91
92          for i = 1:N - 1
93              x(i+1,:) = x(i,:) + dt * x_dot(i,:);
94              y(i+1,:) = y(i,:) + dt * y_dot(i,:);
95              x_dot(i+1,:) = c * (x(i+1,:) - x(i+1,:)
                    .^3/3 - y(i+1,:) + z);
96              y_dot(i+1,:) = (x(i+1,:).^2 + d*x(i+1,:) -
                    b*y(i+1,:) + a) / c;
97          end
98
99          % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
100         X = -3:0.05:3;
101         Y1 = X - X.^3/3 + z;
102         Y2 = (X.^2 + d*X + a) / b;
103
104         % Diagram of the results
```

```
105         figure(1);
106         p = plot(x,y,'k',X,Y1,'.b',X,Y2,'.r');
107         set(p,'LineWidth',2);
108         xlim([-3 3]);
109         ylim([-3 3]);
110
111         figure(2);
112         plot(t,x,'r',t,y,'b');
113
114     case 'c'
115         % Initial conditions
116         x(1,:) = [2  -2  -2.5  2.00  0.0200  -0.02];
117         y(1,:) = [2  -2  0.00  -2.0  -0.015  0.015];
118
119         % Given parameters
120         a = 0.2;
121         d = 1.9;
122
123         % Calculation of the unstable node
124         X_USt = fsolve(@F,0,optimset('Display','off'),a
                ,b,d);
125         Y_USt = y_val(X_USt);
126         % Calculation of the eigenvalues for the stable
                node
127         Lambda_USt = R(1,-trace(X_USt,b,c),deter(X_USt,
                b,c,d));
128
129         x_dot(1,:) = c * (x(1,:) - x(1,:).^3/3 - y(1,:)
                + z);
130         y_dot(1,:) = (x(1,:).^2 + d*x(1,:) - b*y(1,:) +
                a) / c;
131
132         for i = 1:N - 1
133             x(i+1,:) = x(i,:) + dt * x_dot(i,:);
134             y(i+1,:) = y(i,:) + dt * y_dot(i,:);
135             x_dot(i+1,:) = c * (x(i+1,:) - x(i+1,:)
                    .^3/3 - y(i+1,:) + z);
136             y_dot(i+1,:) = (x(i+1,:).^2 + d*x(i+1,:) -
                    b*y(i+1,:) + a) / c;
137         end
138
139         % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
140         X = -3:0.05:3;
```

```
141         Y1 = X - X.^3/3 + z;
142         Y2 = (X.^2 + d*X + a) / b;
143 
144         % Diagram of the results
145         figure(1);
146         p = plot(x,y,'k',X,Y1,'.b',X,Y2,'.r');
147         set(p,'LineWidth',2);
148         xlim([-3 3]);
149         ylim([-3 3]);
150 
151         figure(2);
152         plot(t,x,'r',t,y,'b');
153 
154     case 'd'
155         % Initial conditions
156         x(1,:) = [-2.40   0.065   -2.80];
157         y(1,:) = [-0.85   0.000   0.020];
158 
159         % Given parameters
160         a = 0.323;
161         d = 1.99;
162 
163         % Calculation of the stable node
164         X_St = fsolve(@F,-3,optimset('Display','off'),a
                ,b,d);
165         Y_St = y_val(X_St);
166         % Calculation of the eigenvalues for the stable
                node
167         Lambda_St = R(1,-trace(X_St,b,c),deter(X_St,b,c
                ,d));
168 
169         % Calculation of the unstable node
170         X_USt = fsolve(@F,0,optimset('Display','off'),a
                ,b,d);
171         Y_USt = y_val(X_USt);
172         % Calculation of the eigenvalues for the
                unstable node
173         Lambda_USt = R(1,-trace(X_USt,b,c),deter(X_USt,
                b,c,d));
174 
175         % Calculation of the saddle node
176         X_Sa = fsolve(@F,-1,optimset('Display','off'),a
                ,b,d);
```

```
177         Y_Sa = y_val(X_Sa);
178         % Calculation of the eigenvalues for the saddle
                node
179         Lambda_Sa = R(1,-trace(X_Sa,b,c),deter(X_Sa,b,c
                ,d));
180
181         x_dot(1,:) = c * (x(1,:) - x(1,:).^3/3 - y(1,:)
                + z);
182         y_dot(1,:) = (x(1,:).^2 + d*x(1,:) - b*y(1,:) +
                a) / c;
183
184         for i = 1:N - 1
185             x(i+1,:) = x(i,:) + dt * x_dot(i,:);
186             y(i+1,:) = y(i,:) + dt * y_dot(i,:);
187             x_dot(i+1,:) = c * (x(i+1,:) - x(i+1,:)
                    .^3/3 - y(i+1,:) + z);
188             y_dot(i+1,:) = (x(i+1,:).^2 + d*x(i+1,:) -
                    b*y(i+1,:) + a) / c;
189         end
190
191         % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
192         X = -3:0.05:3;
193         Y1 = X - X.^3/3 + z;
194         Y2 = (X.^2 + d*X + a) / b;
195
196         % Diagram of the results
197         figure(1);
198         p = plot(x,y,'k',X,Y1,'.b',X,Y2,'.r');
199         set(p,'LineWidth',2);
200         xlim([-3 2]);
201         ylim([-1 1.5]);
202
203         figure(2);
204         plot(t,x,'r',t,y,'b');
205
206     case 'e'
207         % Initial conditions
208         x_1(1,:) = [2   -2   -2.9   1.00   0.1   -2.9];
209         x_2(1,:) = [2   -2   -2.9   1.00   0.1   -2.9];
210         y_1(1,:) = [2   -1   -1.0   -1.0   0.1   1.50];
211         y_2(1,:) = [2   -1   -1.0   -1.0   0.1   1.50];
212
213         % Given parameters
```

```
214         a1 = 0.4;
215         a2 = 0.6;
216         d = 2.2;
217
218         x_dot_1(1,:) = c * (x_1(1,:) - x_1(1,:).^3/3 -
               y_1(1,:) + z);
219         x_dot_2(1,:) = c * (x_2(1,:) - x_2(1,:).^3/3 -
               y_2(1,:) + z);
220         y_dot_1(1,:) = (x_1(1,:).^2 + d*x_1(1,:) - b*
               y_1(1,:) + a1) / c;
221         y_dot_2(1,:) = (x_2(1,:).^2 + d*x_2(1,:) - b*
               y_2(1,:) + a2) / c;
222
223         for i = 1:N - 1
224             x_1(i+1,:) = x_1(i,:) + dt * x_dot_1(i,:);
225             y_1(i+1,:) = y_1(i,:) + dt * y_dot_1(i,:);
226             x_2(i+1,:) = x_2(i,:) + dt * x_dot_2(i,:);
227             y_2(i+1,:) = y_2(i,:) + dt * y_dot_2(i,:);
228             x_dot_1(i+1,:) = c * (x_1(i+1,:) - x_1(i+1,
                   :).^3/3 - y_1(i+1,:) + z);
229             x_dot_2(i+1,:) = c * (x_2(i+1,:) - x_2(i+1,
                   :).^3/3 - y_2(i+1,:) + z);
230             y_dot_1(i+1,:) = (x_1(i+1,:).^2 + d*x_1(i
                   +1,:) - b*y_1(i+1,:) + a1) / c;
231             y_dot_2(i+1,:) = (x_2(i+1,:).^2 + d*x_2(i
                   +1,:) - b*y_2(i+1,:) + a2) / c;
232         end
233
234         % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
235         X = -3:0.05:3;
236         Y1 = X - X.^3/3 + z;
237         Y2_1 = (X.^2 + d*X + a1) / b;
238         Y2_2 = (X.^2 + d*X + a2) / b;
239
240         % Diagram of the results
241         figure(1);
242         p = plot(x_1,y_1,'k',x_2,y_2,'b',X,Y1,'.r',X,
               Y2_1,'.k',X,Y2_2,'.b');
243         set(p,'LineWidth',2);
244         xlim([-3 3]);
245         ylim([-3 3]);
246
247         figure(2);
```

```
248         plot(t,x_1,'r',t,y_1,'b',t,x_2,'r',t,y_2,'b');
249
250     case 'f'
251         % Initial conditions
252         x_1(1,:) = [2   -2   -2.9   1.00   0.1   -2.9];
253         x_2(1,:) = [2   -2   -2.9   1.00   0.1   -2.9];
254         x_3(1,:) = [2   -2   -2.9   1.00   0.1   -2.9];
255         y_1(1,:) = [2   -1   -1.0   -1.0   0.1   1.50];
256         y_2(1,:) = [2   -1   -1.0   -1.0   0.1   1.50];
257         y_3(1,:) = [2   -1   -1.0   -1.0   0.1   1.50];
258
259         % Given parameters
260         a = 0.2;
261         d1 = 1.7;
262         d2 = 1.8;
263         d3 = 2;
264
265         x_dot_1(1,:) = c * (x_1(1,:) - x_1(1,:).^3/3 -
               y_1(1,:) + z);
266         x_dot_2(1,:) = c * (x_2(1,:) - x_2(1,:).^3/3 -
               y_2(1,:) + z);
267         x_dot_3(1,:) = c * (x_3(1,:) - x_3(1,:).^3/3 -
               y_3(1,:) + z);
268         y_dot_1(1,:) = (x_1(1,:).^2 + d1*x_1(1,:) - b*
               y_1(1,:) + a) / c;
269         y_dot_2(1,:) = (x_2(1,:).^2 + d2*x_2(1,:) - b*
               y_2(1,:) + a) / c;
270         y_dot_3(1,:) = (x_3(1,:).^2 + d3*x_3(1,:) - b*
               y_3(1,:) + a) / c;
271
272         for i = 1:N - 1
273             x_1(i+1,:) = x_1(i,:) + dt * x_dot_1(i,:);
274             x_2(i+1,:) = x_2(i,:) + dt * x_dot_2(i,:);
275             x_3(i+1,:) = x_3(i,:) + dt * x_dot_3(i,:);
276             y_1(i+1,:) = y_1(i,:) + dt * y_dot_1(i,:);
277             y_2(i+1,:) = y_2(i,:) + dt * y_dot_2(i,:);
278             y_3(i+1,:) = y_3(i,:) + dt * y_dot_3(i,:);
279             x_dot_1(i+1,:) = c * (x_1(i+1,:) - x_1(i+1,
                   :).^3/3 - y_1(i+1,:) + z);
280             x_dot_2(i+1,:) = c * (x_2(i+1,:) - x_2(i+1,
                   :).^3/3 - y_2(i+1,:) + z);
281             x_dot_3(i+1,:) = c * (x_3(i+1,:) - x_3(i+1,
                   :).^3/3 - y_3(i+1,:) + z);
```

```
282             y_dot_1(i+1,:) = (x_1(i+1,:).^2 + d1*x_1(i
                    +1,:) - b*y_1(i+1,:) + a) / c;
283             y_dot_2(i+1,:) = (x_2(i+1,:).^2 + d2*x_2(i
                    +1,:) - b*y_2(i+1,:) + a) / c;
284             y_dot_3(i+1,:) = (x_3(i+1,:).^2 + d3*x_3(i
                    +1,:) - b*y_3(i+1,:) + a) / c;
285         end
286
287         % Diagrams of dx/dt=0 & dy/dt=0 (Nullclines)
288         X = -3:0.05:3;
289         Y1 = X - X.^3/3 + z;
290         Y2_1 = (X.^2 + d1*X + a) / b;
291         Y2_2 = (X.^2 + d2*X + a) / b;
292         Y2_3 = (X.^2 + d3*X + a) / b;
293
294         % Diagram of the results
295         figure(1);
296         p = plot(x_1,y_1,'k',x_2,y_2,'b',x_3,y_3,'g',X,
                Y1,'.r',X,Y2_1,'.k',X,Y2_2,'.b',X,Y2_2,'.g')
                ;
297         set(p,'LineWidth',2);
298         xlim([-3 3]);
299         ylim([-3 3]);
300
301         figure(2);
302         plot(t,x_1,'r',t,y_1,'b',t,x_2,'r',t,y_2,'b',t,
                x_3,'r',t,y_3,'b');
303 end
304         % Stable, Unstable and Saddle Nodes
305         if S == 'a'
306             Stable = ['Stable Node: (',num2str(sprintf(
                    '%1.5f',X_St)),',',num2str(sprintf('%1.5
                    f',Y_St)),')'];
307             EigenvalueSt = ['Eigenvalue : (',num2str(
                    sprintf('%1.5f',Lambda_St(1))),',',
                    num2str(sprintf('%1.5f',Lambda_St(2))),'
                    )'];
308             disp(Stable);
309             disp(EigenvalueSt);
310             disp('=====================================');
311         elseif S == 'b'
312             Stable = ['Stable Node    : (',num2str(
                    sprintf('%1.5f',X_St)),',',num2str(
```

```
                sprintf('%1.5f',Y_St)),')'];
313             EigenvalueSt = ['Eigenvalue     : (',num2str
                    (sprintf('%1.5f',Lambda_St(1))),',',
                    num2str(sprintf('%1.5f',Lambda_St(2))),'
                    )'];
314             Unstable = ['Unstable Node : (',num2str(
                    sprintf('%1.5f',X_USt)),',',num2str(
                    sprintf('%1.5f',Y_USt)),')'];
315             EigenvalueUSt = ['Eigenvalue     : (',
                    num2str(sprintf('%1.5f',Lambda_USt(1))),
                    ',',num2str(sprintf('%1.5f',Lambda_USt
                    (2))),')'];
316             Saddle = ['Saddle Node    : (',num2str(
                    sprintf('%1.5f',X_Sa)),',',num2str(
                    sprintf('%1.5f',Y_Sa)),')'];
317             EigenvalueSa = ['Eigenvalue     : (',num2str
                    (sprintf('%1.5f',Lambda_Sa(1))),',',
                    num2str(sprintf('%1.5f',Lambda_Sa(2))),'
                    )'];
318             disp(Stable);
319             disp(EigenvalueSt);
320             disp('==========================================');
321             disp(Unstable);
322             disp(EigenvalueUSt);
323             disp('==========================================');
324             disp(Saddle);
325             disp(EigenvalueSa);
326             disp('==========================================');
327         elseif S == 'c'
328             Unstable = ['Unstable Node : (',num2str(
                    sprintf('%1.5f',X_USt)),',',num2str(
                    sprintf('%1.5f',Y_USt)),')'];
329             EigenvalueUSt = ['Eigenvalue     : (',
                    num2str(sprintf('%1.5f',Lambda_USt(1))),
                    ',',num2str(sprintf('%1.5f',Lambda_USt
                    (2))),')'];
330             disp(Unstable);
331             disp(EigenvalueUSt);
332             disp('==========================================');
333         elseif S == 'd'
334             Stable = ['Stable Node    : (',num2str(
                    sprintf('%1.5f',X_St)),',',num2str(
                    sprintf('%1.5f',Y_St)),')'];
```

```
335             EigenvalueSt = ['Eigenvalue     : (',num2str
                    (sprintf('%1.5f',Lambda_St(1))),',',
                    num2str(sprintf('%1.5f',Lambda_St(2))),'
                    )'];
336             Unstable = ['Unstable Node : (',num2str(
                    sprintf('%1.5f',X_USt)),',',num2str(
                    sprintf('%1.5f',Y_USt)),')'];
337             EigenvalueUSt = ['Eigenvalue     : (',
                    num2str(sprintf('%1.5f',Lambda_USt(1))),
                    ',',num2str(sprintf('%1.5f',Lambda_USt
                    (2))),')'];
338             Saddle = ['Saddle Node    : (',num2str(
                    sprintf('%1.5f',X_Sa)),',',num2str(
                    sprintf('%1.5f',Y_Sa)),')'];
339             EigenvalueSa = ['Eigenvalue     : (',num2str
                    (sprintf('%1.5f',Lambda_Sa(1))),',',
                    num2str(sprintf('%1.5f',Lambda_Sa(2))),'
                    )'];
340             disp(Stable);
341             disp(EigenvalueSt);
342             disp('=====================================');
343             disp(Unstable);
344             disp(EigenvalueUSt);
345             disp('=====================================');
346             disp(Saddle);
347             disp(EigenvalueSa);
348             disp('=====================================');
349         end
350 end
351
352 function Y=y_val(x)
353     Y = x - x^3/3;
354 end
355 function T=trace(x,b,c)
356     T = c - c*x^2 - b/c;
357 end
358
359 function D=deter(x,b,c,d)
360     D = (c - c*x^2)*(-b/c) - (-c)*(1/c)*(2*x + d);
361 end
362
363 function Func=F(x,a,b,d)
364     Func = x - x^3/3 - (x^2 + d*x + a)/b;
```

```
365 end
366
367 function X=R(a,b,c)
368     Delta = b^2 - 4*a*c;
369     X(1) = (-b + sqrt(Delta))/(2*a);
370     X(2) = (-b - sqrt(Delta))/(2*a);
371 end
```

C.3. Integrate-and-Fire model

```
function I_F
clear;
clc;

%% Given Parameters
c = 0;
T = 1;
p = 0;
e = 0.3;

% Time span
t0 = 0;
tf = 2;
dt = 0.001;
t = t0:dt:tf;
N = length(t);

% Initial conditions
ui(1) = 0;
uj(1) = 0.2;

ui_dot(1) = ui(1) + 1;
uj_dot(1) = uj(1) + 1;

for i = 1:N - 1
    if ui(i)>=T
        ui(i+1) = p + c * (ui(i) - T);
        if uj(i)>=T
            uj(i+1) = p + c * (uj(i) - T);
        else
            uj(i+1) = uj(i) + e;
        end
    else
        ui(i+1) = ui(i) + dt * ui_dot(i);
        if uj(i)>=T
            uj(i+1) = p + c * (uj(i) - T);
        else
            uj(i+1) = uj(i) + dt * uj_dot(i);
        end
    end
    ui_dot(i+1) = ui(i+1) + 1;
```

```
42      uj_dot(i+1) = uj(i+1) + 1;
43  end
44
45  % Diagram of the results
46  subplot(2,1,1);
47  p1 = plot(t,ui,'b');
48  set(p1,'LineWidth',2);
49  xlabel('t','fontsize',14);
50  ylabel('u_i_(_t_)','fontsize',14);
51  xlim([0 2]);
52  ylim([0 1]);
53
54  subplot(2,1,2);
55  p1 = plot(t,uj,'g');
56  set(p1,'LineWidth',2);
57  xlabel('t','fontsize',14);
58  ylabel('u_j_(_t_)','fontsize',14);
59  xlim([0 2]);
60  ylim([0 1]);
61  % set(gca,'fontsize',18);
62  % xlim([-3 3]);
63  % ylim([-3 3]);
64  end
```

C.4. Izhikevich neurons model

```
1
2  %============Neuro-Computational Features============
3
4  %   This MATLAB file generates figure 1 in the paper by
5  %                Izhikevich E.M. (2004)
6  %   Which Model to Use For Cortical Spiking Neurons?
7  %   use MATLAB R13 or later. November 2003. San Diego,
       CA
8
9  %%%%%%%%%%%%%% (A) tonic spiking
       %%%%%%%%%%%%%%%%%%%%%%%
10 subplot(5,4,1)
11 a=0.02; b=0.2;  c=-65;  d=6;
12 V=-70;  u=b*V;
13 VV=[];  uu=[];
14 tau = 0.25; tspan = 0:tau:100;
15 T1=tspan(end)/10;
16 for t=tspan
17     if (t>T1)
18         I=14;
19     else
20         I=0;
21     end;
22     V = V + tau*(0.04*V^2+5*V+140-u+I);
23     u = u + tau*a*(b*V-u);
24     if V > 30
25         VV(end+1)=30;
26         V = c;
27         u = u + d;
28     else
29         VV(end+1)=V;
30     end;
31     uu(end+1)=u;
32 end;
33 plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
34 axis([0 max(tspan) -90 30])
35 axis off;
36 title('(A) tonic spiking');
37
38 %%%%%%%%%%%%%%%%%% (B) phasic spiking
       %%%%%%%%%%%%%%%%%%%%%%%%
```

```
39 subplot(5,4,2)%
40 a=0.02; b=0.25; c=-65;  d=6;
41 V=-64; u=b*V;
42 VV=[]; uu=[];
43 tau = 0.25;tspan = 0:tau:200;
44 T1=20;
45 for t=tspan
46     if (t>T1)
47         I=0.5;
48     else
49         I=0;
50     end;
51     V = V + tau*(0.04*V^2+5*V+140-u+I);
52     u = u + tau*a*(b*V-u);
53     if V > 30
54         VV(end+1)=30;
55         V = c;
56         u = u + d;
57     else
58         VV(end+1)=V;
59     end;
60     uu(end+1)=u;
61 end;
62 plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
63 axis([0 max(tspan) -90 30])
64 axis off;
65 title('(B) phasic spiking');
66
67 %%%%%%%%%%%%% (C) tonic bursting
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
68 subplot(5,4,3)
69 a=0.02; b=0.2;  c=-50;  d=2;
70 V=-70;  u=b*V;
71 VV=[]; uu=[];
72 tau = 0.25; tspan = 0:tau:220;
73 T1=22;
74 for t=tspan
75     if (t>T1)
76         I=15;
77     else
78         I=0;
79     end;
80     V = V + tau*(0.04*V^2+5*V+140-u+I);
```

```
81      u = u + tau*a*(b*V-u);
82      if V > 30
83          VV(end+1)=30;
84          V = c;
85          u = u + d;
86      else
87          VV(end+1)=V;
88      end;
89      uu(end+1)=u;
90  end;
91  plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
92  axis([0 max(tspan) -90 30])
93  axis off;
94  title('(C) tonic bursting');
95
96  %%%%%%%%%%%%%% (D) phasic bursting
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%
97  subplot(5,4,4)
98  a=0.02; b=0.25; c=-55;  d=0.05;
99  V=-64;  u=b*V;
100 VV=[];  uu=[];
101 tau = 0.2;  tspan = 0:tau:200;
102 T1=20;
103 for t=tspan
104     if (t>T1)
105         I=0.6;
106     else
107         I=0;
108     end;
109     V = V + tau*(0.04*V^2+5*V+140-u+I);
110     u = u + tau*a*(b*V-u);
111     if V > 30
112         VV(end+1)=30;
113         V = c;
114         u = u + d;
115     else
116         VV(end+1)=V;
117     end;
118     uu(end+1)=u;
119 end;
120 plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
121 axis([0 max(tspan) -90 30])
122 axis off;
```

```
123 title('(D) phasic bursting');
124
125
126 %%%%%%%%%%%%%% (E) mixed mode
       %%%%%%%%%%%%%%%%%%%%%%%%%%
127 subplot(5,4,5)
128 a=0.02; b=0.2;  c=-55;  d=4;
129 V=-70;  u=b*V;
130 VV=[];  uu=[];
131 tau = 0.25; tspan = 0:tau:160;
132 T1=tspan(end)/10;
133 for t=tspan
134     if (t>T1)
135         I=10;
136     else
137         I=0;
138     end;
139     V = V + tau*(0.04*V^2+5*V+140-u+I);
140     u = u + tau*a*(b*V-u);
141     if V > 30
142         VV(end+1)=30;
143         V = c;
144         u = u + d;
145     else
146         VV(end+1)=V;
147     end;
148     uu(end+1)=u;
149 end;
150 plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
151 axis([0 max(tspan) -90 30])
152 axis off;
153 title('(E) mixed mode');
154
155
156 %%%%%%%%%%%%%% (F) spike freq. adapt
       %%%%%%%%%%%%%%%%%%%%%%%%%%
157 subplot(5,4,6)
158 a=0.01; b=0.2;  c=-65;  d=8;
159 V=-70;  u=b*V;
160 VV=[];  uu=[];
161 tau = 0.25; tspan = 0:tau:85;
162 T1=tspan(end)/10;
163 for t=tspan
```

```
164     if (t>T1)
165         I=30;
166     else
167         I=0;
168     end;
169     V = V + tau*(0.04*V^2+5*V+140-u+I);
170     u = u + tau*a*(b*V-u);
171     if V > 30
172         VV(end+1)=30;
173         V = c;
174         u = u + d;
175     else
176         VV(end+1)=V;
177     end;
178     uu(end+1)=u;
179 end;
180 plot(tspan,VV,[0 T1 T1 max(tspan)],-90+[0 0 10 10]);
181 axis([0 max(tspan) -90 30])
182 axis off;
183 title('(F) spike freq. adapt');
184
185 %%%%%%%%%%%%%%%% (G) Class 1 exc.
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%
186 subplot(5,4,7)
187 a=0.02; b=-0.1; c=-55; d=6;
188 V=-60; u=b*V;
189 VV=[]; uu=[];
190 tau = 0.25; tspan = 0:tau:300;
191 T1=30;
192 for t=tspan
193     if (t>T1)
194         I=(0.075*(t-T1));
195     else
196         I=0;
197     end;
198     V = V + tau*(0.04*V^2+4.1*V+108-u+I);
199     u = u + tau*a*(b*V-u);
200     if V > 30
201         VV(end+1)=30;
202         V = c;
203         u = u + d;
204     else
205         VV(end+1)=V;
```

```
206     end;
207     uu(end+1)=u;
208 end;
209 plot(tspan,VV,[0 T1 max(tspan) max(tspan)],-90+[0 0 20
      0]);
210 axis([0 max(tspan) -90 30])
211 axis off;
212 title('(G) Class 1 excitable');
213
214 %%%%%%%%%%%%%%%%% (H) Class 2 exc.
      %%%%%%%%%%%%%%%%%%%%%%%%%%
215 subplot(5,4,8)
216 a=0.2;  b=0.26; c=-65;  d=0;
217 V=-64;  u=b*V;
218 VV=[];  uu=[];
219 tau = 0.25; tspan = 0:tau:300;
220 T1=30;
221 for t=tspan
222     if (t>T1)
223         I=-0.5+(0.015*(t-T1));
224     else
225         I=-0.5;
226     end;
227     V = V + tau*(0.04*V^2+5*V+140-u+I);
228     u = u + tau*a*(b*V-u);
229     if V > 30
230         VV(end+1)=30;
231         V = c;
232         u = u + d;
233     else
234         VV(end+1)=V;
235     end;
236     uu(end+1)=u;
237 end;
238 plot(tspan,VV,[0 T1 max(tspan) max(tspan)],-90+[0 0 20
      0]);
239 axis([0 max(tspan) -90 30])
240 axis off;
241 title('(H) Class 2 excitable');
242
243 %%%%%%%%%%%%%%%%% (I) spike latency
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%
244 subplot(5,4,9)
```

```
245 a=0.02; b=0.2;  c=-65;  d=6;
246 V=-70;  u=b*V;
247 VV=[];  uu=[];
248 tau = 0.2; tspan = 0:tau:100;
249 T1=tspan(end)/10;
250 for t=tspan
251     if t>T1 & t < T1+3
252         I=7.04;
253     else
254         I=0;
255     end;
256     V = V + tau*(0.04*V^2+5*V+140-u+I);
257     u = u + tau*a*(b*V-u);
258     if V > 30
259         VV(end+1)=30;
260         V = c;
261         u = u + d;
262     else
263         VV(end+1)=V;
264     end;
265     uu(end+1)=u;
266 end;
267 plot(tspan,VV,[0 T1 T1 T1+3 T1+3 max(tspan)],-90+[0 0
       10 10 0 0]);
268 axis([0 max(tspan) -90 30])
269 axis off;
270 title('(I) spike latency');
271
272
273 %%%%%%%%%%%%%%%% (J) subthresh. osc.
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%
274 subplot(5,4,10)
275 a=0.05; b=0.26; c=-60;  d=0;
276 V=-62;  u=b*V;
277 VV=[];  uu=[];
278 tau = 0.25; tspan = 0:tau:200;
279 T1=tspan(end)/10;
280 for t=tspan
281     if (t>T1) & (t < T1+5)
282         I=2;
283     else
284         I=0;
285     end;
```

```
286     V = V + tau*(0.04*V^2+5*V+140-u+I);
287     u = u + tau*a*(b*V-u);
288     if V > 30
289         VV(end+1)=30;
290         V = c;
291         u = u + d;
292     else
293         VV(end+1)=V;
294     end;
295     uu(end+1)=u;
296 end;
297 plot(tspan,VV,[0 T1 T1 (T1+5) (T1+5) max(tspan)],-90+[0
        0 10 10 0 0],...
298       tspan(220:end),-10+20*(VV(220:end)-mean(VV)));
299 axis([0 max(tspan) -90 30])
300 axis off;
301 title('(J) subthreshold osc.');
302
303
304 %%%%%%%%%%%%%%%%%% (K) resonator
       %%%%%%%%%%%%%%%%%%%%%%%%%
305 subplot(5,4,11)
306 a=0.1;  b=0.26; c=-60;  d=-1;
307 V=-62;  u=b*V;
308 VV=[];  uu=[];
309 tau = 0.25; tspan = 0:tau:400;
310 T1=tspan(end)/10;
311 T2=T1+20;
312 T3 = 0.7*tspan(end);
313 T4 = T3+40;
314 for t=tspan
315     if ((t>T1) & (t < T1+4)) | ((t>T2) & (t < T2+4)) |
          ((t>T3) & (t < T3+4)) | ((t>T4) & (t < T4+4))
316         I=0.65;
317     else
318         I=0;
319     end;
320     V = V + tau*(0.04*V^2+5*V+140-u+I);
321     u = u + tau*a*(b*V-u);
322     if V > 30
323         VV(end+1)=30;
324         V = c;
325         u = u + d;
```

```
326     else
327         VV(end+1)=V;
328     end;
329     uu(end+1)=u;
330 end;
331 plot(tspan,VV,[0 T1 T1 (T1+8) (T1+8) T2 T2 (T2+8) (T2
      +8) T3 T3 (T3+8) (T3+8) T4 T4 (T4+8) (T4+8) max(
      tspan)],-90+[0 0 10 10 0 0 10 10 0 0 10 10 0 0 10 10
       0 0]);
332 axis([0 max(tspan) -90 30])
333 axis off;
334 title('(K) resonator');
335
336 %%%%%%%%%%%%%% (L) integrator
      %%%%%%%%%%%%%%%%%%%%%%%%
337 subplot(5,4,12)
338 a=0.02; b=-0.1; c=-55; d=6;
339 V=-60; u=b*V;
340 VV=[]; uu=[];
341 tau = 0.25; tspan = 0:tau:100;
342 T1=tspan(end)/11;
343 T2=T1+5;
344 T3 = 0.7*tspan(end);
345 T4 = T3+10;
346 for t=tspan
347     if ((t>T1) & (t < T1+2)) | ((t>T2) & (t < T2+2)) |
        ((t>T3) & (t < T3+2)) | ((t>T4) & (t < T4+2))
348         I=9;
349     else
350         I=0;
351     end;
352     V = V + tau*(0.04*V^2+4.1*V+108-u+I);
353     u = u + tau*a*(b*V-u);
354     if V > 30
355         VV(end+1)=30;
356         V = c;
357         u = u + d;
358     else
359         VV(end+1)=V;
360     end;
361     uu(end+1)=u;
362 end;
```

```
363 plot(tspan,VV,[0 T1 T1 (T1+2) (T1+2) T2 T2 (T2+2) (T2
      +2) T3 T3 (T3+2) (T3+2) T4 T4 (T4+2) (T4+2) max(
      tspan)],-90+[0 0 10 10 0 0 10 10 0 0 10 10 0 0 10 10
       0 0]);
364 axis([0 max(tspan) -90 30])
365 axis off;
366 title('(L) integrator');
367
368 %%%%%%%%%%%%%%%% (M) rebound spike
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
369 subplot(5,4,13)
370 a=0.03; b=0.25; c=-60;  d=4;
371 V=-64;  u=b*V;
372 VV=[];  uu=[];
373 tau = 0.2;  tspan = 0:tau:200;
374 T1=20;
375 for t=tspan
376     if (t>T1) & (t < T1+5)
377         I=-15;
378     else
379         I=0;
380     end;
381     V = V + tau*(0.04*V^2+5*V+140-u+I);
382     u = u + tau*a*(b*V-u);
383     if V > 30
384         VV(end+1)=30;
385         V = c;
386         u = u + d;
387     else
388         VV(end+1)=V;
389     end;
390     uu(end+1)=u;
391 end;
392 plot(tspan,VV,[0 T1 T1 (T1+5) (T1+5) max(tspan)],-85+[0
      0 -5 -5 0 0]);
393 axis([0 max(tspan) -90 30])
394 axis off;
395 title('(M) rebound spike');
396
397 %%%%%%%%%%%%%%%% (N) rebound burst
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
398 subplot(5,4,14)
399 a=0.03; b=0.25; c=-52;  d=0;
```

```
V=-64;  u=b*V;
VV=[];  uu=[];
tau = 0.2;  tspan = 0:tau:200;
T1=20;
for t=tspan
    if (t>T1) & (t < T1+5)
        I=-15;
    else
        I=0;
    end;
    V = V + tau*(0.04*V^2+5*V+140-u+I);
    u = u + tau*a*(b*V-u);
    if V > 30
        VV(end+1)=30;
        V = c;
        u = u + d;
    else
        VV(end+1)=V;
    end;
    uu(end+1)=u;
end;
plot(tspan,VV,[0 T1 T1 (T1+5) (T1+5) max(tspan)],-85+[0
    0 -5 -5 0 0]);
axis([0 max(tspan) -90 30])
axis off;
title('(N) rebound burst');

%%%%%%%%%%%%%%%% (O) thresh. variability
   %%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(5,4,15)
a=0.03; b=0.25; c=-60;  d=4;
V=-64;  u=b*V;
VV=[];  uu=[];
tau = 0.25; tspan = 0:tau:100;
for t=tspan
   if ((t>10) & (t < 15)) | ((t>80) & (t < 85))
        I=1;
    elseif (t>70) & (t < 75)
        I=-6;
    else
        I=0;
    end;
    V = V + tau*(0.04*V^2+5*V+140-u+I);
```

```
441     u = u + tau*a*(b*V-u);
442     if V > 30
443         VV(end+1)=30;
444         V = c;
445         u = u + d;
446     else
447         VV(end+1)=V;
448     end;
449     uu(end+1)=u;
450 end;
451 plot(tspan,VV,[0 10 10 15 15 70 70 75 75 80 80 85 85
       max(tspan)],...
452             -85+[0 0  5  5  0  0  -5 -5 0  0  5  5  0  0]
                  );
453 axis([0 max(tspan) -90 30])
454 axis off;
455 title('(O) thresh. variability');
456
457
458 %%%%%%%%%%%%%% (P) bistability
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%
459 subplot(5,4,16)
460 a=0.1;  b=0.26; c=-60;  d=0;
461 V=-61;  u=b*V;
462 VV=[];  uu=[];
463 tau = 0.25; tspan = 0:tau:300;
464 T1=tspan(end)/8;
465 T2 = 216;
466 for t=tspan
467     if ((t>T1) & (t < T1+5)) | ((t>T2) & (t < T2+5))
468         I=1.24;
469     else
470         I=0.24;
471     end;
472     V = V + tau*(0.04*V^2+5*V+140-u+I);
473     u = u + tau*a*(b*V-u);
474     if V > 30
475         VV(end+1)=30;
476         V = c;
477         u = u + d;
478     else
479         VV(end+1)=V;
480     end;
```

```
481     uu(end+1)=u;
482 end;
483 plot(tspan,VV,[0 T1 T1 (T1+5) (T1+5) T2 T2 (T2+5) (T2
      +5) max(tspan)],-90+[0 0 10 10 0 0 10 10 0 0]);
484 axis([0 max(tspan) -90 30])
485 axis off;
486 title('(P) bistability');
487
488
489 %%%%%%%%%%%%%% (Q) DAP %%%%%%%%%%%%%%%%%%%%%%%%%%%
490 subplot(5,4,17)
491 a=1;  b=0.2; c=-60;  d=-21;
492 V=-70;  u=b*V;
493 VV=[];  uu=[];
494 tau = 0.1; tspan = 0:tau:50;
495 T1 = 10;
496 for t=tspan
497      if abs(t-T1)<1
498          I=20;
499     else
500          I=0;
501     end;
502     V = V + tau*(0.04*V^2+5*V+140-u+I);
503     u = u + tau*a*(b*V-u);
504     if V > 30
505         VV(end+1)=30;
506         V = c;
507         u = u + d;
508     else
509         VV(end+1)=V;
510     end;
511     uu(end+1)=u;
512 end;
513 plot(tspan,VV,[0 T1-1 T1-1 T1+1 T1+1 max(tspan)],-90+[0
      0 10 10 0 0]);
514 axis([0 max(tspan) -90 30])
515 axis off;
516 title('(Q) DAP          ');
517
518
519
520 %%%%%%%%%%%%%% (R) accomodation
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
521 subplot(5,4,18)
522 a=0.02;  b=1; c=-55;  d=4;
523 V=-65;  u=-16;
524 VV=[];  uu=[];  II=[];
525 tau = 0.5; tspan = 0:tau:400;
526 for t=tspan
527     if (t < 200)
528         I=t/25;
529     elseif t < 300
530         I=0;
531     elseif t < 312.5
532         I=(t-300)/12.5*4;
533     else
534         I=0;
535     end;
536     V = V + tau*(0.04*V^2+5*V+140-u+I);
537     u = u + tau*a*(b*(V+65));
538     if V > 30
539         VV(end+1)=30;
540         V = c;
541         u = u + d;
542     else
543         VV(end+1)=V;
544     end;
545     uu(end+1)=u;
546     II(end+1)=I;
547 end;
548 plot(tspan,VV,tspan,II*1.5-90);
549 axis([0 max(tspan) -90 30])
550 axis off;
551 title('(R) accomodation');
552
553 %%%%%%%%%%%%% (S) inhibition induced spiking
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%
554 subplot(5,4,19)
555 a=-0.02;  b=-1; c=-60;  d=8;
556 V=-63.8;  u=b*V;
557 VV=[];  uu=[];
558 tau = 0.5; tspan = 0:tau:350;
559 for t=tspan
560        if (t < 50) | (t>250)
561         I=80;
562     else
```

```
563         I=75;
564     end;
565     V = V + tau*(0.04*V^2+5*V+140-u+I);
566     u = u + tau*a*(b*V-u);
567     if V > 30
568         VV(end+1)=30;
569         V = c;
570         u = u + d;
571     else
572         VV(end+1)=V;
573     end;
574     uu(end+1)=u;
575 end;
576 plot(tspan,VV,[0 50 50 250 250 max(tspan)],-80+[0 0 -10
    -10 0 0]);
577 axis([0 max(tspan) -90 30])
578 axis off;
579 title('(S) inh. induced sp.');
580
581 %%%%%%%%%%%%% (T) inhibition induced bursting
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
582 subplot(5,4,20)
583 a=-0.026;  b=-1; c=-45;  d=-2;
584 V=-63.8;  u=b*V;
585 VV=[];  uu=[];
586 tau = 0.5; tspan = 0:tau:350;
587 for t=tspan
588         if (t < 50) | (t>250)
589          I=80;
590     else
591         I=75;
592     end;
593     V = V + tau*(0.04*V^2+5*V+140-u+I);
594     u = u + tau*a*(b*V-u);
595     if V > 30
596         VV(end+1)=30;
597         V = c;
598         u = u + d;
599     else
600         VV(end+1)=V;
601     end;
602     uu(end+1)=u;
603 end;
```

```
604 plot(tspan,VV,[0 50 50 250 250 max(tspan)],-80+[0 0 -10
        -10 0 0]);
605 axis([0 max(tspan) -90 30])
606 axis off;
607 title('(T) inh. induced brst.');
608 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
609
610 set(gcf,'Units','normalized','Position',[0.3 0.1 0.6
       0.8]);
611
612
613 %==========Spiking Neurons==========
614
615  function
616 %created by M. Izhikevich, feb 25, 2003
617 %   ecitatory neurons  inhibitory neurons
618 clear;
619 clc;
620
621
622 %% Given Parameter
623 Ne=800;
624 Ni=200;
625 re=rand(Ne,1);
626 ri=rand(Ni,1);
627 a=[0.02*ones(Ne,1); 0.02+0.08*ri];
628 b=[0.2*ones(Ne,1); 0.25-0.05*ri];
629 c=[-65+15*re.^2; -65*ones(Ni,1)];
630 d=[8-6*re.^2; 2*ones(Ni,1)];
631 S=[0.5*rand(Ne+Ni,Ne), -rand(Ne+Ni,Ni)];
632
633 v=-65*ones(Ne+Ni,1); % Initial values of v
634 u=b.*v; % Initial values of u
635 firings=[]; % spike timings
636 for t=1:1000 % simulation of 1000 ms
637 I=[5*randn(Ne,1);2*randn(Ni,1)]; % thalamic input
638 fired=find(v>=30); % indices of spikes
639 firings=[firings; t+0*fired,fired];
640 v(fired)=c(fired);
641 u(fired)=u(fired)+d(fired);
642 I=I+sum(S(:,fired),2);
643 v=v+0.5*(0.04*v.^2+5*v+140-u+I); % step 0.5 ms
644 v=v+0.5*(0.04*v.^2+5*v+140-u+I); % for numerical
```

```
645 u=u+a.*(b.*v-u); % stability
646 end;
647 plot(firings(:,1),firings(:,2),'.');
648 end
```

C.5. Hodgkin-Huxley model

```
%% Hodgkin_Huxley
function HH

I = 0.0;%15;%0.35;
C = 1;
GNa = 120;
GK = 36;
%Gm = 0.3;
GL = 0.3;
ENa = 120;%115;%50;
EK = -12; %-77;%-12;
EL = 10.6;%-54.4;
Vrest = 0;%10.613;

%% Initial Conditions

t0 = 0;
tf = 10;
dt = 0.001;
tspan = t0:dt:tf;
v = -40;%-30
m = 0;
n = 0;
h = 1;
V = [];
M = [];
N = [];
H = [];
T_M = [];
T_N = [];
T_H = [];
M_INF = [];
N_INF = [];
H_INF = [];


% N = length(t);

for t=tspan

```

```
41      v = v + (dt/C)*(I - GK*n^4*(v - EK) - GNa*m^3*h*(v
           - ENa) - GL*(v - EL));
42      %v = v + (dt/C)*(GNa*m^3*h*(ENa-v) + Gk*n^4*(Ek-v)
           + Gm*(Vrest - v) + I);
43
44      alpha_m = (0.1 * (25 - v)) / (exp((25 - v)/10) - 1)
           ;
45      %alpha_m = (0.1 * (v + 40)) / (1 - exp(-(v+40)/10))
           ;
46      %alpha_m = (25 - v) /10*(exp((25 - v)/10) - 1);
47      %beta_m = 4* exp(-v/18);
48      %beta_m = 4 * exp(-(v+65)/18);
49      beta_m = 4 * exp((-v)/18);
50      m = m + dt * (alpha_m*(1-m) - beta_m*m);
51      t_m = 1 / (alpha_m + beta_m);
52
53      alpha_n = (0.01 * (10 - v)) / (exp((10 - v)/10) -
           1);
54      %alpha_n = (0.01 * (v + 55)) / (1 - exp(-(v+55)/10)
           );
55      %beta_n = 0.125 * exp(-V/80);
56      %alpha_n = (10 - v) / 100 * (exp((10 - v)/10) - 1);
57      %beta_n = 0.125 * exp(-(v+65)/80);
58      beta_n = 0.125 * exp(-v/80);
59      n = n + dt * (alpha_n*(1-n) - beta_n*n);
60      t_n = 1 / (alpha_n + beta_n);
61
62      alpha_h = 0.07 * exp(-v/20);
63      %alpha_h = 0.07 * exp(-(v+65)/20);
64      %alpha_h = 0.07 * exp(-v/20);
65      %beta_h = 1 / (exp((30 - v)/10) + 1);
66      %beta_h = 1 / (1 + exp(-(v+35)/10));
67      beta_h = 1/ (exp((30 - v)/10) + 1);
68      h = h + dt * (alpha_h*(1-h) - beta_h*h);
69      t_h = 1 / (alpha_h + beta_h);
70
71      m_inf = alpha_m / (alpha_m + beta_m);
72      n_inf = alpha_n / (alpha_n + beta_n);
73      h_inf = alpha_h / (alpha_h + beta_h);
74
75
76      V(end+1) = v;
77      M(end+1) = m;
```

```
78      N(end+1) = n;
79      H(end+1) = h;
80      T_M(end+1) = t_m;
81      T_N(end+1) = t_n;
82      T_H(end+1) = t_h;
83      M_INF(end+1) = m_inf;
84      N_INF(end+1) = n_inf;
85      H_INF(end+1) = h_inf;
86
87
88  end
89
90
91
92
93          figure(1);
94          p = plot(tspan,V,'b');
95          set(p,'LineWidth',2);
96          xlim([0 10]);
97          Y = ylabel('V [mV]');
98          X = xlabel('Time [ms]');
99          xlabel('Time [ms]','fontsize',20);
100         ylabel('V [mV]','fontsize',20);
101 %         set(X,'FontWeight','Bold');
102 %         set(Y,'FontWeight','Bold');
103         set(gca,'fontsize',20);
104
105         figure(2);
106         p = plot(tspan,M,'g',tspan,N,'k',tspan,H,'m');
107         set(p,'LineWidth',2);
108         xlim([0 10]);
109         Y = ylabel('m,n,h');
110         X = xlabel('Time [ms]');
111         xlabel('Time [ms]','fontsize',20);
112         ylabel('m,n,h','fontsize',20);
113 %         set(X,'FontWeight','Bold');
114 %         set(Y,'FontWeight','Bold');
115         set(gca,'fontsize',20);
116
117 %         figure(3);
118 %         plot(V,M,'g',V,N,'k',V,H,'m');
119 %         Y = ylabel('m,n,h');
120 %         X = xlabel('V [mV]');
```

```
121 %           xlabel('V [mV]','fontsize',18);
122 %           ylabel('m,n,h','fontsize',18);
123 %           set(gca,'fontsize',18);
124
125           figure(4);
126           p = plot(V,T_M,'g',V,T_N,'k',V,T_H,'m');
127           set(p,'LineWidth',2);
128           xlim([-40 100]);
129           Y = ylabel('t_m,t_n,t_h');
130           X = xlabel('V [mV]');
131           xlabel('V [mV]','fontsize',20);
132           ylabel('\tau_m,\tau_n,\tau_h','fontsize',20);
133 %           set(X,'FontWeight','Bold');
134 %           set(Y,'FontWeight','Bold');
135           set(gca,'fontsize',20);
136
137           figure (5);
138           p = plot(V,M_INF,'g',V,N_INF,'k',V,H_INF,'m');
139           set(p,'LineWidth',2);
140           xlim([-40 100]);
141           Y = ylabel('m_{inf},n_{inf},h_{inf}');
142           X = xlabel('V [mV]');
143           xlabel('V [mV]','fontsize',20);
144           ylabel('m\infty,n\infty,h\infty','fontsize',20)
                ;
145 %           set(X,'FontWeight','Bold');
146 %           set(Y,'FontWeight','Bold');
147           set(gca,'fontsize',20);
148
149
150
151 end
```

C.6. Morris-Lecar model

```
1  %% Morris Lecar
2  function ML_2(S)
3  S = 1;
4  Cm = 20;
5  gleak = 2;
6  gK = 8;
7  vK = -84;
8  vleak = -60;
9  vCa = 120;
10 v_1 = -1.2;
11 v_2 = 18;
12
13 %% Initial Conditions
14
15 t0 = 0;
16 tf = 200;
17 dt = 0.01;
18 t = t0:dt:tf;
19 N = length(t);
20
21 switch S;
22     % Homoclinic Bifurcation (1) and Homoclinic Regime
          (2-1 and 2-2 with diff. start values) at I = I0
23     case 1
24         gCa = 4;
25         v_3 = 12;
26         v_4 = 17.4;
27         fi = 0.23;
28
29         I0_1 = 35; %for Homoclinic Bifurcation
30         I0_2 = 39.5; %for Homoclinic Regime
31
32         %initial conditions for Homoclinic Bifurcation
33         v1(1) = -28.6;
34         W1(1) = -0.1;
35
36         %initial conditions for Homoclinic Regime
37         v2_1(1) = -32;
38         v2_2(1) = -5;
39         W2_1(1) = -0.09;
40         W2_2(1) = -0.1;
```

```
41
42          for i=1:N-1
43              M1(i) = 1/2 * (1 + tanh((v1(i) - v_1) / v_2
                    ));
44              M2_1(i) = 1/2 * (1 + tanh((v2_1(i) - v_1) /
                     v_2));
45              M2_2(i) = 1/2 * (1 + tanh((v2_2(i) - v_1) /
                     v_2));
46
47              WW1(i) = 1/2 * (1 + tanh((v1(i) - v_3) /
                    v_4));
48              WW2_1(i) = 1/2 * (1 + tanh((v2_1(i) - v_3)
                    / v_4));
49              WW2_2(i) = 1/2 * (1 + tanh((v2_2(i) - v_3)
                    / v_4));
50
51              tau1(i) = 1 / cosh((v1(i) - v_3) / (2 * v_4
                    ));
52              tau2_1(i) = 1 / cosh((v2_1(i) - v_3) / (2 *
                     v_4));
53              tau2_2(i) = 1 / cosh((v2_2(i) - v_3) / (2 *
                     v_4));
54
55              f1_1 = (1/Cm) * (I0_1 - gleak*(v1(i)-vleak)
                     - gK*W1(i)*(v1(i)-vK) - gCa*M1(i)*(v1(i
                    )-vCa));
56              f1_2_1 = (1/Cm) * (I0_2 - gleak*(v2_1(i)-
                    vleak) - gK*W2_1(i)*(v2_1(i)-vK) - gCa*
                    M2_1(i)*(v2_1(i)-vCa));
57              f1_2_2 = (1/Cm) * (I0_2 - gleak*(v2_2(i)-
                    vleak) - gK*W2_2(i)*(v2_2(i)-vK) - gCa*
                    M2_2(i)*(v2_2(i)-vCa));
58
59              f2_1 = fi * (WW1(i)-W1(i)) / tau1(i);
60              f2_2_1 = fi * (WW2_1(i)-W2_1(i)) / tau2_1(i
                    );
61              f2_2_2 = fi * (WW2_2(i)-W2_2(i)) / tau2_2(i
                    );
62
63              v1(i+1) = v1(i) + dt * f1_1;
64              v2_1(i+1) = v2_1(i) + dt * f1_2_1;
65              v2_2(i+1) = v2_2(i) + dt * f1_2_2;
66
```

```
67              W1(i+1) = W1(i) + dt * f2_1;
68              W2_1(i+1) = W2_1(i) + dt * f2_2_1;
69              W2_2(i+1) = W2_2(i) + dt * f2_2_2;
70          end
71
72
73          figure(1);
74          p = plot(v1,W1,'b');
75          Y = ylabel('W');
76          X = xlabel('V [mV]');
77          set(X,'FontWeight','Bold');
78          set(Y,'FontWeight','Bold');
79          set(p,'LineWidth',1.5);
80          xlabel('V [mV]','fontsize',18);
81          ylabel('W','fontsize',18);
82          set(gca,'fontsize',18);
83          xlim([-40 24]);
84          ylim([-0.1 0.5]);
85
86          figure(2);
87          p = plot(t,v1);
88          set(p,'LineWidth',2);
89          xlabel('t','FontSize',20);
90          ylabel('V','FontSize',20);
91          set(gca,'fontsize',18);
92
93          figure(3);
94          p = plot(t,W1);
95          set(p,'LineWidth',2);
96          xlabel('t','FontSize',20);
97          ylabel('W','FontSize',20);
98          set(gca,'fontsize',18);
99
100         figure(4);
101         p = plot(v2_1,W2_1,'b',v2_2,W2_2,'b');
102         Y = ylabel('W');
103         X = xlabel('V [mV]');
104         set(X,'FontWeight','Bold');
105         set(Y,'FontWeight','Bold');
106         set(p,'LineWidth',1.5);
107         xlabel('V [mV]','fontsize',18);
108         ylabel('W','fontsize',18);
109         set(gca,'fontsize',18);
```

```
110         xlim([-40 24]);
111         ylim([-0.1 0.5]);
112
113         figure(5);
114         p = plot(t,v2_1);
115         set(p,'LineWidth',2);
116         xlabel('t','FontSize',20);
117         ylabel('V','FontSize',20);
118         set(gca,'fontsize',18);
119
120         figure(6);
121         p = plot(t,W2_1);
122         set(p,'LineWidth',2);
123         xlabel('t','FontSize',20);
124         ylabel('W','FontSize',20);
125         set(gca,'fontsize',18);
126
127
128
129     % Homoclinic Regime I = I0
130     case 2
131         I0 = 39.5;
132         gCa = 4;
133         v_3 = 12;
134         v_4 = 17.4;
135         fi = 0.23;
136         v(1) = -28.6;
137         W(1) = -0.1;
138         time = 5;
139
140         for i=1:N-1
141             M(i) = 1/2 * (1 + tanh((v(i) - v_1) / v_2))
                    ;
142             WW(i) = 1/2 * (1 + tanh((v(i) - v_3) / v_4)
                    );
143             tau(i) = 1 / cosh((v(i) - v_3) / (2 * v_4))
                    ;
144
145             f1 = (1/Cm) * (I0 - gleak*(v(i)-vleak) - gK
                    *W(i)*(v(i)-vK) - gCa*M(i)*(v(i)-vCa));
146             f2 = fi * (WW(i)-W(i)) / tau(i);
147
148             v(i+1) = v(i) + dt * f1;
```

```
149                 W(i+1) = W(i) + dt * f2;
150             end
151 end
```

Index