Universidad del País Vasco
Euskal Herriko Unibertsitatea

DOCTORAL THESIS

"A mathematical, computational and experimental study of neuronal excitability"

by Guillaume Girier

Doctoral school of computer engineering
at the University of the Basque Country
(UPV/EHU, Leioa, Spain)

Under the supervision of:

Serafim Rodrigues, BCAM, Ikerbasque.

During the academic years:

2020 - 2024

Collaborator :

Mathieu Desroches, INRIA, France.

In collaboration with :

A mathematical, computational and experimental study of the neuronal excitability

Reviewers
Daniele Avitabile, Associate Prof., Faculty of Science, Mathematics, VU Amsterdam, Amsterdam, The Netherlands.
Denis Sheynikhovich, Associate Prof. and head of the "Aging in Vision and Action" lab at the *Vision Institute*, Sorbonne Université, Paris, France.

Examiners
Daniele Avitabile, Associate Prof., Faculty of Science, Mathematics, VU Amsterdam, Amsterdam, The Netherlands.
Manuel Maria Graña Romay, Prof., Department of Computer Science and Artificial Intelligence, UPV EHU, Leioa, Spain.
Jan Tønnesen, Group leader, CSIC titular, Nanoscale Neurophysiology Lab, Instituto Biofisika, Leioa, Spain.

Substitutes
Denis Sheynikhovich, Associate Prof. and head of the "Aging in Vision and Action" lab at the *Vision Institute*, Sorbonne Université, Paris, France.
Ana Gonzalez Acuña, Prof., Department of Computer Science and Artificial Intelligence, UPV EHU, Leoia, Spain.
Catalina Vich Llompart, Senior Lecturer, Department de Ciènces Matemàtiques i Informàtica, Universitat de les Illes Balears, Palma, Spain.

Une étude mathématique, informatique et expérimentale de l'excitabilité neuronale

---

Résumé

L'excitabilité neuronale fait référence à la capacité des neurones à générer des signaux électriques, appelés potentiels d'action, en réponse à des stimuli. Ce concept peut être étudié sous différents aspects (mathématiques, informatiques et expérimentaux). Dans cette thèse, nous nous intéresserons à étudier ce concept en superposant, lorsque nécessaire, ces différents aspects afin d'en extraire de nouveaux résultats, et ce à travers cinq projets différents. Dans le premier projet, nous étudierons d'abord mathématiquement la transition comportementale entre les neurones intégrateurs (modèles de neurones de type I) et les neurones résonateurs (modèles neurones de type II) dans des modèles mathématiques de neurones tout en conservant les propriétés qui font du modèle un neurone intégrateur. Dans un deuxième projet, nous analyserons les données neuronales obtenues à partir des enregistrements *patch-clamp* de cellules Granule (GC) au cours de leur développement. Au cours d'une période transitoire de maturation, les propriétés intrinsèques et synaptiques des nouveaux GC présentent des propriétés distinctes de celles des GC matures, sous-tendant potentiellement la contribution de la neurogenèse au codage de la mémoire. Nous produirons un modèle adapté à ce comportement. Dans deux projets liés, nous nous concentrerons sur l'obtention de diagrammes de bifurcation à partir d'expériences bruitées, avec des méthodes inspirées de la continuation numérique, appelées *Control Based Continuation in Experiments* (CBCE). L'idée est d'appliquer un contrôle en boucle fermée à une expérience et de rendre le contrôle itératif non invasif, ce qui révèle l'attracteur de l'expérience non contrôlée. Dans le dernier projet, nous analyserons les données d'imagerie calcique du bulbe olfactif de plusieurs souris auxquelles différents produits chimiques ont été présentés. Notre objectif principal était de mettre en évidence des schémas de réactions neuronales aux stimuli, mais également de développer un pipeline permettant de comparer l'activité de différents sujets, à travers des cartes odotopiques.

Mots clés

Neurosciences computationnelles, Mathématiques, Informatique, Neurosciences expérimentales, Excitabilité neuronale, Bifurcations, Modélisation, Analyse de données, Analyse de structure de récurrence.

A mathematical, computational and experimental study of the neuronal excitability

---

Abstract

Neuronal excitability refers to the ability of neurons to generate electrical signals, called action potentials, in response to stimuli. This concept can be studied through different aspects (mathematical, computational and experimental). In this thesis, we will be interested in studying this concept by overlapping, when necessary, these different aspects in order to extract new results, and this through five different projects. In the first project, we will first mathematically study the behavioral transition between integrator neurons (type-I neuron models) and resonator neurons (type-II neuron models) in mathematical models of neurons while retaining the properties that make the model an integrator neuron. In a second project, we will analyze neuronal data obtained from *patch-clamp* recordings of Granule cells (GC) during their development. During a transient period of maturation, new GCs intrinsic and synaptic properties exhibit distinct from mature GCs, potentially underlying the contribution of neurogenesis to memory encoding. We will produce a model adapted to this behavior. In two related projects, we will focus on obtaining bifurcation diagrams from noisy experiments, with methods inspired by digital continuation, called *Control Based Continuation in Experiments* (CBCE). The idea is apply closed-loop control to an experiment and iteratively bring the control to being noninvasive, which reveals the attractor of the uncontrolled experiment. In the last project, we will analyze calcium imaging data from the olfactory bulb of several mice to which different chemicals were presented. Our main objective was to highlight neural reaction patterns to stimuli, but also to develop a pipeline allowing to compare the activity of different subjects, through odotopic maps.

Keywords

Computational neuroscience, Mathematics, Computer science, Experimental neuroscience, Neural excitability, Bifurcations, Modeling, Data analysis, Recurrence structure analysis.

Un estudio matemático, computacional y experimental de la excitabilidad neuronal

---

Resumen obligatorio de mínimo 5 páginas (solicitud UPV)

La excitabilidad neuronal se refiere a la capacidad de las neuronas para generar señales eléctricas, llamadas potenciales de acción (PA), en respuesta a estímulos. Las neuronas son células excitables que pueden responder a una variedad de estímulos, incluidas señales químicas o eléctricas de otras células, cambios en la concentración iónica en su entorno o estímulos mecánicos como la presión. La excitabilidad neuronal puede ser modulada por una variedad de factores, incluidos neurotransmisores, hormonas, cambios en la concentración de iones en el entorno de la neurona y la presencia o ausencia de estímulos inhibidores. Las variaciones en la excitabilidad neuronal son esenciales para el funcionamiento normal del cerebro y del sistema nervioso, y pueden estar implicadas en una variedad de trastornos neurológicos y psiquiátricos. El estudio de la excitabilidad neuronal es clave para comprender el funcionamiento del cerebro, cómo se comunican las neuronas entre sí y cómo se forman y modifican los circuitos neuronales en respuesta a estímulos ambientales.

Existen varios modelos de neuronas individuales que nos permiten estudiar la noción de umbral de excitabilidad: en los años 1940 y 1950, Alan Lloyd Hodgkin y Andrew Huxley desarrollaron un modelo matemático (HH) de la AP en las neuronas [1] (que obtuvo Premio Nobel en 1962) basado en experimentos sobre la iniciación y propagación de potenciales de acción en el axón gigante del calamar. El modelo HH describe los mecanismos subyacentes a la propagación de AP a lo largo del axón de una neurona [2, 3], mediante el uso de ecuaciones diferenciales. Gracias a sus experimentos y modelo, también desentrañaron el papel clave de los canales Na^+ y K^+ en la generación de AP y la propagación de señales. Por lo tanto, el modelo HH fue un punto de inflexión importante en el estudio matemático de la excitabilidad neuronal porque permitió a los científicos predecir con precisión las características de los AP y comprender cómo los diferentes tipos de canales iónicos contribuyen a la generación y propagación de los AP.

El modelo de FitzHugh-Nagumo (FHN) es un sistema de dos ecuaciones diferenciales ordinarias que ha jugado un papel fundamental en el modelado matemático de la actividad eléctrica de las neuronas y en la comprensión de los mecanismos subyacentes a las AP neuronales [4, 5]. Este enfoque, desarrollado de forma independiente por dos investigadores, Richard FitzHugh en 1961 y Jinichi Nagumo en 1962, ofrece una simplificación del modelo HH descrito anteriormente y del complejo comportamiento de las neuronas, preservando al mismo tiempo las características esenciales de su dinámica.

En 1981, el modelo Morris-Lecar (ML) fue desarrollado por Catherine Morris y Harold Lecar [6]. Es un modelo matemático de la actividad eléctrica de las neuronas. Este modelo se basa en el modelo HH, sigue el mismo formalismo y, por tanto, los movimientos de los iones a través de la membrana. Este modelo fue desarrollado para describir la actividad eléctrica de las células de las glándulas del estómago de la sepia, pero desde entonces se ha utilizado para estudiar la actividad eléctrica de otros tipos de neuronas. Debido a su dimensión reducida (2D), es un modelo relativamente simple comparado con otros modelos más complejos, como el modelo de Hodgkin-Huxley, lo que lo hace más fácil de entender e implementar. El modelo se ha utilizado para estudiar muchos aspectos de la actividad eléctrica de las neuronas, como la regulación de la excitabilidad neuronal, las propiedades electrofisiológicas de los canales iónicos, la plasticidad sináptica y las oscilaciones neuronales.

Estos tres modelos neuronales serán los utilizados a lo largo de esta tesis. Esta tesis propone un estudio en profundidad de la excitabilidad neuronal a través de diferentes enfoques, que van desde sistemas dinámicos hasta experimentos de laboratorio con exploraciones computacionales. Cada capítulo contiene resultados novedosos sobre la excitabilidad neuronal, así como la validación de modelos matemáticos como representaciones fieles de experimentos.

En el capítulo 2, estudiamos la transición de comportamiento entre las neuronas integradoras (modelos de neuronas tipo I) y las neuronas resonadoras (neuronas tipo II) en modelos matemáticos de neuronas. Los sistemas excitables se pueden clasificar según la existencia de oscilaciones por debajo del umbral. Esta característica divide todos los sistemas en dos tipos de neuronas: integradora o resonadora. Las neuronas integradoras se definen por: a) la capacidad de excitarse bajo pulsos de alta frecuencia, b) la existencia de un umbral preciso y c) el hecho de que no tienen una oscilación por debajo del umbral. Son, por tanto, neuronas de excitabilidad de clase I (lo que significa que el régimen AP comienza en frecuencia cero). Por el contrario, las resonadoras son neuronas que: a) responden sólo a pulsos en intervalos de frecuencia bien definidos, b) no tienen un umbral específico, y c) tienen oscilaciones por debajo del umbral. Por lo tanto, pertenecen a la clase II (lo que significa que el régimen AP comienza con una frecuencia distinta de cero). En un marco fisiológico, las neuronas se comportan como integradoras o como resonadoras, y esto no al mismo tiempo.

En este capítulo, proponemos una nueva estrategia para obtener un cambio de excitabilidad en un modelo neuronal, basada en la presencia de múltiples escalas de tiempo en este modelo. Ilustramos nuestro enfoque con el llamado modelo I_{Na}/I_K de E.M. Izhikevich, estudiado bajo el efecto de una corriente externa sinusoidal lenta. Esto crea una estructura lenta-rápida caracterizada por la presencia en el espacio de fases de un punto especial llamado *singularidad en silla plegada*. Utilizando resultados recientes sobre la teoría geométrica de la perturbación singular relacionada con las sillas plegadas, modificamos el forzamiento lento para obtener condiciones algebraicas en la silla plegada que se sabe que crean oscilaciones de pequeña amplitud en las trayectorias que pasan por ella en el espacio de fase. En el contexto neuronal, mostramos que estas oscilaciones de pequeña amplitud están por debajo del umbral y, por lo tanto, dotan al modelo integrador de las características de un resonador. Este efecto no depende de la elección del modelo de tipo I, solo requiere la estructura de bifurcación asociada (SNIC) de un modelo de tipo I. También mostramos un escenario para el cambio inverso, del resonador al integrador, nuevamente basado en dinámicas de múltiples escalas de tiempo, pero esta vez dependiente de una *singularidad de nodo plegado*. Se sabe que los nodos plegados están asociados con oscilaciones de pequeña amplitud que, en el contexto neuronal, están por debajo del umbral. Concluimos sugiriendo un protocolo experimental, basado en la configuración *dynamic-clamp* en electrofisiología, para probar nuestras predicciones teóricas y proponer también algunas perspectivas sobre la relevancia de nuestros hallazgos. Los resultados de este capítulo aparecieron recientemente en la revista *Nonlinear Dynamics* (IF 5.6) en un artículo titulado *From integrator to resonator neurons: A multiple-timescale scenario*[1] [7].

Una neurona inmadura es una neurona que aún no ha logrado la diferenciación a una forma madura. Durante el desarrollo neuronal, las neuronas inmaduras sufren muchas transformaciones hasta alcanzar su estado maduro. Por ejemplo, deben establecer conexiones sinápticas con otras neuronas para formar redes neuronales funcionales, y también deben desarrollar dendritas y axones para facilitar la transmisión de señales eléctricas y químicas. Las neuronas inmaduras juegan

[1] doi: 10.1007/s11071-023-08687-1.

un papel importante en la configuración del cerebro en desarrollo, ya que tienen la capacidad de dividirse y diferenciarse en diferentes tipos de neuronas especializadas. Este proceso se llama *neurogénesis* y es crucial para la formación de circuitos neuronales funcionales. Sin embargo, ciertos tipos de neuronas inmaduras también pueden estar presentes en los adultos, como las células madre neurales en determinadas regiones del cerebro. Estas células tienen la capacidad de dividirse y diferenciarse en diferentes tipos de neuronas y células gliales, lo que puede ayudar a reparar el daño cerebral o regenerar el tejido dañado.

El Capítulo 3 se centró en el análisis de datos neuronales obtenidos de registros *patch-clamp* de células granulares (GC) inmaduras durante su desarrollo. La excitabilidad de estas neuronas puede variar según su etapa de desarrollo y su ubicación en el cerebro. En general, tienen mayor excitabilidad que las neuronas maduras debido a la presencia de canales iónicos inmaduros en su membrana celular. Estos canales iónicos inmaduros pueden permitir una mayor entrada de sodio en las neuronas inmaduras, haciendo que su potencial de membrana sea más positivo y facilitando la generación de AP, los impulsos eléctricos utilizados para transmitir información a través del sistema nervioso. Con el tiempo, las neuronas inmaduras sufren cambios morfológicos y funcionales hasta alcanzar su estado maduro. Los canales iónicos maduros reemplazan a los canales inmaduros, lo que reduce la entrada de sodio y aumenta la eficiencia de los circuitos neuronales.

Durante un período transitorio de maduración, los nuevos GC exhiben propiedades intrínsecas y sinápticas distintas de los GC maduros, lo que potencialmente subyace a la contribución de la neurogénesis a la codificación de la memoria [8–14]. En este capítulo, buscamos explorar la transición de los GC en maduración al bloqueo de despolarización mediante el uso de abrazadera dinámica. Comprender el mecanismo preciso del bloqueo de la despolarización en los GC puede conducir a una mejor comprensión de los cambios electrofisiológicos que experimentan los GC durante el proceso de maduración. Mi parte del trabajo en este proyecto fue modelar los datos recopilados y su excitabilidad anormal. Para igualar este comportamiento, utilizamos un modelo tipo Hodgkin-Huxley en el que el canal de sodio tiene más estados de lo normal. Los resultados de este proyecto se han recogido en un manuscrito titulado *Complex excitability and "flipping" of granule cells: an experimental and computational study* que se encuentra actualmente en revisión[2] [15].

Una neurona tiene diferentes regímenes (estado de reposo, régimen de picos) y, por lo tanto, hay transiciones entre estos regímenes que, en los modelos matemáticos subyacentes, corresponden a *bifurcaciones* [16]. La bifurcación es un concepto clave en matemáticas que tiene como objetivo describir cómo los sistemas dinámicos responden a cambios en los parámetros y cómo se producen transiciones cualitativas en su comportamiento cuando estos parámetros varían. Esta teoría se centra en identificar puntos críticos, llamados bifurcaciones, donde el comportamiento del sistema sufre una transición fundamental. Estas bifurcaciones pueden resultar en cambios en las trayectorias, atractores, períodos u otras características del sistema. Las curvas de atractores y repelentes, ya sean estacionarias o periódicas, separadas por puntos de bifurcación y representadas en un mismo plano, forman *diagramas de bifurcación de un parámetro*.

En los capítulos 4 y 5, nos centramos en obtener diagramas de bifurcación a partir de experimentos ruidosos y métodos de continuación numérica [17, 18]. Basado en un control de retroalimentación de bucle cerrado, con un algoritmo de búsqueda de raíces incorporado (por

[2] doi: 10.13140/RG.2.2.34294.57921.

ejemplo, el método de Newton), el método de continuación de experimentos basado en control (CBCE) permite superar el hecho de que no conocemos a priori las ecuaciones rectoras asociadas con el experimento que estamos realizando. estudiando. En el capítulo 4, estudiamos datos experimentales obtenidos al someter una neurona a dos protocolos diferentes de electrofisiología de abrazadera de parche: una abrazadera de voltaje con rampa lenta en la tensión de retención (VC) y una abrazadera de corriente con rampa lenta en la corriente de retención (CC). El resultado del protocolo VC proporcionó una curva (corriente, voltaje) que se asemeja a un diagrama de bifurcación en estado estacionario. Es más, una vez superpuestos al resultado del protocolo CC, los dos conjuntos de datos interactuaron de manera similar a una solución de sistemas completos superpuestos a un diagrama de bifurcación de estado estable de subsistema rápido. Es decir, un efecto de disección lento-rápido. En este capítulo, explicamos por qué funciona este método simplificado y planteamos hipótesis sobre la cuestión del ruido obtenido alrededor de las partes inestables de la variedad crítica. Así, este método, que es similar a una revisión simplificada del CBCE (sin la parte de Newton), nos permite obtener una aproximación viable de un diagrama de bifurcación experimental en estado estacionario, y también puede validar el modelo neuronal subyacente como una buena representación de la verdadera neurona. Los resultados de este proyecto se han recopilado en un manuscrito titulado *Observing hidden neuronal states in experiments* que se encuentra actualmente en revisión[3] [19].

En el capítulo 5, nos centramos en aplicar una versión completa del método CBCE en experimentos simulados mediante Simulink. Simulink es un software de modelado y simulación gráfica desarrollado por MathWorks. Se utiliza en muchos campos de la ingeniería, incluida la neurociencia computacional, para simular y analizar sistemas dinámicos. La operación en tiempo real de Simulink permite la simulación de sistemas en tiempo real mediante el uso de algoritmos en tiempo real para controlar los procesos de simulación. Esto permite probar sistemas y controladores en un entorno virtual antes de implementarlos en el mundo real. Los beneficios de usar Simulink incluyen la capacidad de visualizar sistemas fácilmente, agregar y editar bloques para refinar modelos, depurar modelos en tiempo real y trabajar con herramientas de diseño gráfico fáciles de usar.

La versión completa del CBCE se acerca más a resultados precisos a la hora de obtener curvas de puntos estacionarios. En cuanto a las ramas de soluciones periódicas, el método puede calcularlas fácilmente cuando las oscilaciones son ”suficientemente simples”, es decir, en modelos de tipo forma normal. Sin embargo, cuando las oscilaciones son más no lineales como en el modelo de FitzHugh-Nagumo, el método parece tener más problemas para aproximar la rama correcta.

La recurrencia es una propiedad fundamental de muchos sistemas dinámicos; fue formulada por primera vez por Poincaré [20] y es omnipresente en diversos procesos en la naturaleza y en los sistemas biológicos. Un sistema puede divergir fuertemente, pero después de un tiempo se repite infinitas veces tan cerca como se desee de su estado inicial. En otras palabras, determinadas regiones del espacio de estados (del sistema asociado) son visitadas frecuentemente a lo largo del tiempo. Eckmann *et al.* [21] introdujo por primera vez una visualización de esta recurrencia y se pueden utilizar diferentes medidas para estudiar las propiedades del sistema. Sin embargo, se utiliza un parámetro de umbral fundamental, ε, para identificar y determinar el nivel de observación de la recurrencia. Para proporcionar una observación óptima, es posible optimizar el valor de ε, lo que permite analizar con mayor precisión los estados transitorios y metaestables (es decir, estados dinámicos cuasi-invariantes) de un sistema [22–25]. Se ha demostrado la aplicabilidad

[3] doi: 10.48550/arXiv.2308.15477.

a la dinámica cerebral. [24]. Específicamente, el método RSA identifica una condición óptima (función de utilidad parametrizada por ε) basada en una matriz estocástica. La función de utilidad se construye mediante la traza de la matriz estocástica y la entropía de las columnas y filas de la matriz, y finalmente se maximiza mediante un método de optimización, que determina el ε óptimo. Esto luego permite revelar los estados recurrentes (dados como estados transitorios cuasi-invariantes o estados metaestables) de un sistema dado.

En el capítulo 6, presentamos un estudio analítico de datos sobre el bulbo olfatorio de ratones. Es decir, analizamos datos de imágenes de calcio procedentes de 3 ratones sometidos a diferentes sustancias químicas. Desarrollamos un pipeline para estudiar los datos: primero segmentamos el espacio de coordenadas real de las neuronas con el método k-means, luego produjimos el promedio de las neuronas de cada segmento para cada individuo. Luego, procesamos previamente los datos con el método de alineación de referencia y finalmente aplicamos el método RSA. Estos primeros resultados nos mostraron que efectivamente existen patrones de recurrencia de estos reconocimientos olfativos. A continuación, haber segmentado los datos en subgrupos según las coordenadas de las neuronas en el bulbo olfatorio nos permite comparar los centroides de estos estados metaestables. Estos resultados se resumirán en mapas odotópicos que demuestran que las mismas áreas del bulbo olfatorio se activan por la misma sustancia química. Además, también demostraremos que para los diferentes tipos de sustancias químicas se utilizan las mismas áreas para el reconocimiento pero en diferentes niveles de intensidad. Finalmente propondremos el aprendizaje de un clasificador supervisado de estos centroides para demostrar que son significativamente suficientes para discriminarlos.

En el Apéndice 8, proponemos el estudio de un circuito neuromórfico basado en el modelo de Morris-Lecar. Un circuito neuromórfico es un tipo de circuito electrónico diseñado para reproducir el funcionamiento de las redes neuronales en el cerebro humano, esto incluye la simulación de la electrofisiología neuronal, como la generación de potenciales de acción, transmisión sináptica y otros procesos eléctricos y químicos vinculados a las neuronas. actividad. El interés de estudiar un circuito neuromórfico de una sola neurona para esta tesis era tener un experimento modelado en el que conozcamos las ecuaciones que guían el comportamiento del modelo. Lo que esperábamos era poder aplicar nuestras metodologías desarrolladas durante la tesis en este circuito para calibrarlas, pero también trabajar en la noción de experiencia en tiempo real, que era un nuevo desafío para estos métodos. En la configuración de bucle abierto, el experimento funcionó bien; sin embargo, con la configuración de bucle cerrado encontramos problemas que no se pudieron solucionar. Es decir, retrasos entre la aplicación de la corriente dada al experimento y la lectura de la salida.

Por tanto, esta tesis demostró que el estudio de la excitabilidad neuronal mediante una combinación de enfoques matemáticos, computacionales y experimentales es crucial para comprender el funcionamiento del sistema nervioso. Las matemáticas proporcionan el marco esencial para modelar el complejo comportamiento de las neuronas, permitiéndonos generar representaciones abstractas y predicciones cuantitativas que guían nuestros experimentos. La computación mejora esta capacidad al proporcionarnos herramientas sofisticadas para simular y analizar modelos y explorar conjuntos de datos. Los experimentos proporcionan datos valiosos y tangibles que fundamentan nuestra comprensión en la realidad biológica. La combinación de estos enfoques sigue siendo un tema de investigación en curso, con nuevos resultados disponibles, que nos permitirán contribuir a la comprensión de la actividad neuronal. El método CBCE es un ejemplo de contribución como forma de cerrar la brecha entre modelos y experimentos, y revelar estados neuronales inestables y su papel.

Palabras clave
Neurociencia computacional, Matemáticas, Informática, Experimental, Excitabilidad neuronal, Análisis de datos, Análisis de estructuras de recurrencia, Diagrama de bifurcación, Modelado.

Acknowledgments

I would like to thank Mathieu Desroches for having faith in me, for giving me the opportunity to start a career in computational neuroscience during my Master's programme, and for introducing me to Serafim Rodrigues. I also thank Serafim Rodrigues for giving me the opportunity to join his team at BCAM and for trusting me to develop the MCEN lab. Of course, I thank them for supervising my thesis during these three years full of adventures. I would also like to thank Bruno Delord, who gave me a new taste for mathematics and advised me to follow this path 7 years ago.

I would also like to thank all the collaborators I had throughout this thesis and who led to successful projects: Peter beim Graben, Jan Sieber, Anton Chizhov, Joanna Danielewicz, Tobias Ackels, Andreas Schaefer and Dmitry Amakhin.

I would also like to thank the Bambinos for supporting me for three years, Anne-Carmen Sanchez for her help with certain aspects of data analysis, Rémi Calvin for his listening and his time, María Gabriela Saldaña for her patience and affection, Marinela Dishkova for my well-being in Bilbao, Park Jihyo for her smile and Jean-Philippe Roux for his ability to make people laugh in all circumstances.

Merci. Thank you. Gracias.

Contents

Acronyms

AP Action potential. 2

CBCE Control-based continuation in experiments. 9

CC Current clamp. 52

DRS Desingularized Reduced System. 22

FHN FitzHugh-Nagumo model. 3

GSPT Geometric singular perturbation theory. 32

HB Hopf bifurcation point. 6

HH Hodgkin-Huxley model. 3

KNN K-Nearest Neighbour. 91

LP Limit point (or saddle-node) bifurcation. 67

ML Morris-Lecar model. 3

MMO Mixed-mode oscillations. 33

RP Recurrence plot. 12

RSA Recurrence Structure Analysis. 12

SGD Stochastic Gradient Descent. 91

SMOTE Synthetic Minority Over-sampling Technique. 91

SNIC Saddle-node on invariant circle. 5

SVM Support Vector Machine. 91

VC Voltage clamp. 52

Chapter 1

Introduction

1.1 General objectives

First of all, we propose to describe what the main objectives of the thesis will be, and an overview of what will be presented. As the title of the thesis indicates, we will therefore focus on studying neuronal excitability through different methods from the mathematical, computational and experimental fields. The keystone of this work will be to combine these three fields, when possible, in order to obtain a more complete understanding of neuronal excitability. All the concepts named in this section will be described in the following sections.

Some chapters will be based on mathematical methods, such as the *slow-fast dynamical systems theory* [26–28], in order to study, understand or modify the excitability of neurons, and *in fine*, to produce experimental protocols applicable to real neurons. It will therefore be essential to pay attention to the limits of what is possible to apply through experimental setups. In Chapter 2, we will propose to study models of integrator or resonator neurons [29–31], and the possibility of changing the behavior of one type of neuron to obtain the behavior of the second type, without changing its mathematical characteristics; and this using an experimentally applicable protocol.

In Chapter 3, we will study the transient behavior of a granule neuron which passes from the immature stage to the mature stage. This behavior obtained through an experimental protocol designed by collaborators will be the subject of the construction of a model which simulates this same protocol. This project will therefore have the main objective of studying the excitability portrait of a maturing dendate gyrus neuron through a model established for this occasion.

Other chapters will be based on both the computational and experimental aspects of the thesis: in fact, beyond simulating mathematical models, we will also contribute in (Chapters 4 and 5) bridging the gap between dynamical models and real neuronal recording. Indeed, mathematical models can often be considered as simple examples of neural mechanisms and behaviors. We will therefore seek to go further, by adapting a methodology used in the field of differential equations to real experiments in order to obtain *one-parameter experimental bifurcation diagrams* [17, 18]. These kinds of representation will allow us to observe the excitability threshold of the neuron, the spiking regions, but also the unstable states of a neuron.

From a computational perspective, Chapter 6 will be about analyzing data from the olfactory bulb of mice that are stimulated with chemicals. Indeed, the notion of neuronal excitability can be studied through experimental data in order to understand in more detail the neuronal activation patterns, or the regions of interest in the assimilation of a stimulus. This chapter therefore aims to study neuronal excitability through data analytical methods.

Still with the aim of bridging the gap between dynamical models, and real neuronal recording, in Appendix A, we showcase the work that we have been carried out on a neuromorphic circuit: an electronic circuit designed to reproduce the behavior of a neuron model. This includes the generation of action potentials, or even synaptic transmission. The interest of such a chapter was therefore to study a “controlled” experiment in order to apply the methodologies developed during the thesis. However, difficulties were encountered during the development of the project, but as this work is in the same direction as the thesis, we have decided to include it as an appendix. The

importance of studying this type of experiment, and the limitations encountered during the thesis are also discussed in the appendix.

In summary, the main objectives of the whole thesis are to: 1) bring together different fields of research with the aim of studying neuronal excitability, 2) to bridge the gap between mathematics and the experimental world, via the computational aspect, in order to use differential equation models and associated methodologies as tools for studying neuron recordings, 3) to show the diversity of possible approaches to studying the notion of neuronal excitability.

In the following section, the different concepts, methods and experimental protocols will be described in detail in order to give the reader a better understanding. Then, a detailed roadmap for the organization of the thesis will be proposed.

1.2 Neuronal Excitability

1.2.1 Biological background

Neuronal excitability refers to the ability of neurons to generate electrical signals, called action potentials (AP), in response to stimuli. Neurons are excitable cells, which can respond to a variety of stimuli, including chemical or electrical signals from other cells, changes in ionic concentration in their environment, or mechanical stimuli such as pressure.

When a neuron is stimulated, an imbalance sets in between sodium (Na^+) and potassium (K^+) ions and it can reach a excitability threshold that triggers the generation of an AP. An AP is an electrical signal that travels down the neuron's axon and can trigger the release of neurotransmitters from the neuron's synaptic terminals. These neurotransmitters can then excite or inhibit other neurons with which the original neuron is connected. This phenomenon is therefore of capital importance in the transmission of information. Moreover, the principle of "none or all" suggests that the transmission of information between two neurons occurs in a binary manner, which means that either the signal is transmitted entirely or it is not transmitted at all. Thus, the idea is that as long as the necessary excitability threshold is not reached, there will be no response from the postsynaptic neuron. A small amount of neurotransmitters will not cause a partial response from the neuron, but it will take a large enough amount to trigger a full response. This ensures that information is transmitted reliably and accurately, which is essential for the functioning of the nervous system. (See Fig. 1.1)

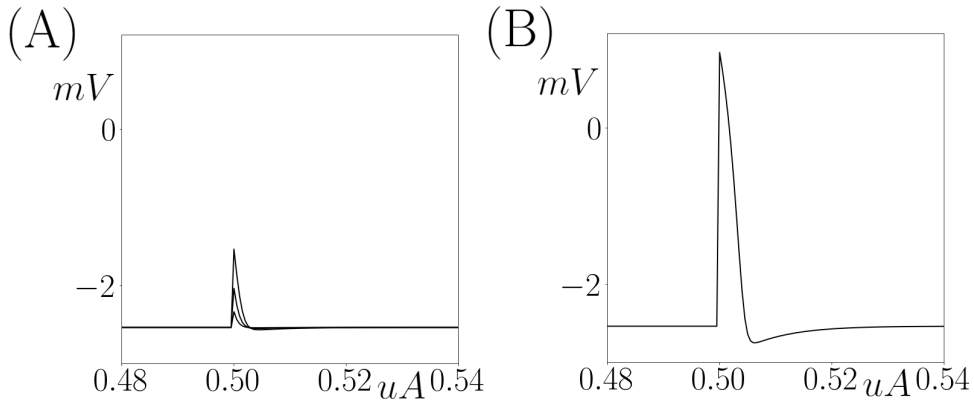

Figure 1.1: "All or none" effect: (A) The stimulation bring the membrane potential below the excitability threshold, then the neuron returns to the resting state. (B) The excitability threshold is reached and an AP is therefore generated.

Neuronal excitability can be modulated by a variety of factors, including neurotransmitters, hormones, changes in ion concentration in the environment of the neuron, and the presence or

absence of inhibitory stimuli. Variations in neuronal excitability are essential for normal brain and nervous system function, and may be implicated in a variety of neurological and psychiatric disorders. The study of neuronal excitability is hence important to understand the functioning of the brain and how neurons communicate with each other and how neural circuits are formed and modified in response to environmental stimuli.

There are various single neuron models that allow us to study this notion of excitability threshold: for instance, the Integrate-and-Fire model (also called leaky integrate-and-fire), is one of the simplest models for describing the electrical behavior of neuron [32]. It was first investigated in 1907 by Louis Lapicque and makes it possible to study in particular the capacity of a neuron to integrate and transmit electrical signals. However, it is only a phenomelogical model and it does not take into account some more complex aspects of neuronal biology, such as specific ion channels, details of AP propagation.

In the 1940s and 1950s, Alan Lloyd Hodgkin and Andrew Huxley developed a mathematical model (HH) of the AP in neurons [1] (which earned them a Nobel Prize in 1962) based on experiments about initiation and propagation of action potentials in the squid giant axon. The HH model describe so the mechanims underlying the propagation of AP along the axon of a neuron [2, 3], by using differential equations. Thanks to their experiments and model, they also unraveled the key role of Na^+ and K^+ channels in the AP generation and signal propagation. Hence, the HH model was a major turning point in the mathematical study of neuronal excitability because it allowed scientists to accurately predict the characteristics of APs and understand how different types of ion channels contribute generation and propagation of APs.

The FitzHugh-Nagumo model (FHN) is a system of two ordinary differential equations that has played a fundamental role in the mathematical modeling of the electrical activity of neurons and in understanding the mechanisms underlying neuronal APs [4, 5]. This approach, developed independently by two researchers, Richard FitzHugh in 1961 and Jinichi Nagumo in 1962, offers a simplification of the HH model described previously, and of the complex behavior of neurons while preserving the essential characteristics of their dynamics.

In 1981, the Morris-Lecar model (ML) was developed by Catherine Morris and Harold Lecar [6]. It is a mathematical model of the electrical activity of neurons. This model is based on the HH model follows the same formalism and so the movements of ions across the membrane. This model was developed to describe the electrical activity of cuttlefish stomach gland cells, but it has since been used to study the electrical activity of other types of neurons. Due to its reduced dimension (2D), it is a relatively simple model compared to other more complex models, such as the Hodgkin-Huxley model, which makes it easier to understand and implement. The model has been used to study many aspects of neuron electrical activity, such as regulation of neuronal excitability, electrophysiological properties of ion channels, synaptic plasticity, and neuronal oscillations.

As part of this thesis, we will use some of these models which we will describe in more depth in each of the associated chapters.

When we study a mathematical model like those described above, the rest state of the neuron actually corresponds to a stable state, from which we deviate by disturbing the stability of the model. Thus, like what we have seen on the notion of all-or-none, a weak disturbance, through the current imposed on the neuron which does not allow the excitability threshold to be reached, will return the steady state solution. Once the threshold has been passed, then we obtain an AP.

Alan Lloyd Hodgkin also defined three classes of neurons distinguished by their frequency-current (f–I) curves [33]. This classification is so based on the dynamics of the APs of neurons modeled using differential equations:

1) The **type I neurons** are characterized by an appearance of APs at a low frequency, depending on the strength of the applied current

2) The **type II neurons** are characterized by an appearance of APs at a certain frequency that is relatively insensitive to changes in the strength of the applied current.

3) The **type III neurons** are the neurons which can have only one spike, then return to resting state, no matter what the input is.

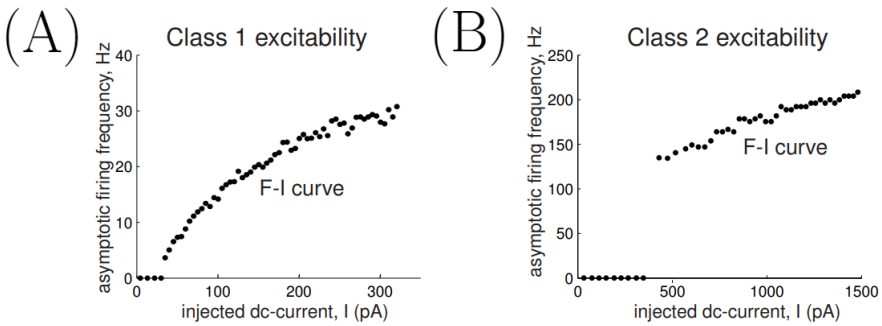

Figure 1.2: Frequency-current (F-I) relations of (A) an class 1 neuron (cortical pyramidal neuron) and (B) a class 2 neuron (brainstem mesV neuron). (Adapted from [34], Fig. 1.14)

Thus, the different classes of neuronal excitability refer to the different ways in which neurons respond to electrical or chemical stimuli. By studying these different classes of neuronal excitability, one can better understand the underlying mechanisms that regulate neuronal activity. This may have important implications for the treatment of certain neurological diseases, such as epilepsy, where abnormal regulation of neuronal excitability can lead to seizures, but most importantly it helps us understand how neurons work and how they communicate. In this thesis, we will focus on the first two classes of excitability.

1.2.2 Mathematical background

The previous section decribes the fact that a neuron has different regimes (resting state, spiking regime) and that there are therefore transitions between these regimes, which, in the underlying mathematical models, correspond to *bifurcations* [16]. Bifurcation is a key concept in mathematics which aims to understand how dynamical systems respond to changes in parameters and how qualitative transitions in their behavior occur when these parameters vary. This theory focuses on identifying critical points, called *bifurcations*, where the behavior of the system undergoes a fundamental transition. These bifurcations can result in changes in the trajectories, attractors, periods, or other characteristics of the system. Curves of attractors and repellors, whether stationary or periodic, separated by bifurcation points and represented in a same plane, form *one-parameter bifurcation diagrams*.

Calculating bifurcation diagram analytically may be quite challenging: the most basic method to obtain them is called the *brute-force approach* and consists in simulating the system for long enough so as to exhaust the transient dynamics and reach a small-enough vicinity of an attractor. It can be a stationary attractor (equilibrium), a periodic attractor (limit cycle), either simple or more complicated (e.g., period-doubled, period-quadrupled cycles, etc.), or even chaotic attractors. The main drawback is that it does not give access to bifurcation points, nor to unstable branches. However, it can be used to identify the presence of chaotic attractors and understand the route to chaos in parameter space.

Fortunately, there are robust numerical algorithms to compute such objects, namely *numerical continuation* methods. One of these important methods is called *natural parameter continuation* (also called *naive continuation*) [35]: this is a predictor-corrector algorithm used to compute the solution set of a (possibly multi-dimensional) algebraic equation which is "under-determined" (that is, with one more unknown than equation) and hence admits a one-parameter family of solutions. Starting from a point on the branch one seeks to compute, the algorithm allows to find subsequent points by using a prediction, which perturbs the parameter and the initial guess away from the previous point; and then, a correction which allows to converge the predicted new

point iteratively back to the correct branch by using a root-finding method, for instance, Newton's method [36]. Naive continuation makes a very simple prediction where only the parameter is perturbed. Nevertheless, this method has a limit: the Newton corrector fail when the jacobian matrix is singular. This phenomena happen when the determinant of this matrix is equal to zero, so when we have a fold bifurcation type.

The most successful method is the one called *Pseudo-arclength continuation* designed by Herbert Keller [35, 37]. The main idea of this method is to reparametrise the solution curve by the arclength, instead of having it parametrised by the main bifurcation parameter. Using this idea, one can go past fold points and compute both stable and unstable branches. This method is the one that we will use throughout the thesis when we study the bifurcation diagrams of ODE neuron models. Most of the time we will use the software XPPAUT [1], which incorporates a module containing the pseudo-arclength continuation package AUTO, implemented by Eusebius J. Doedel. Doedel [38, 39].

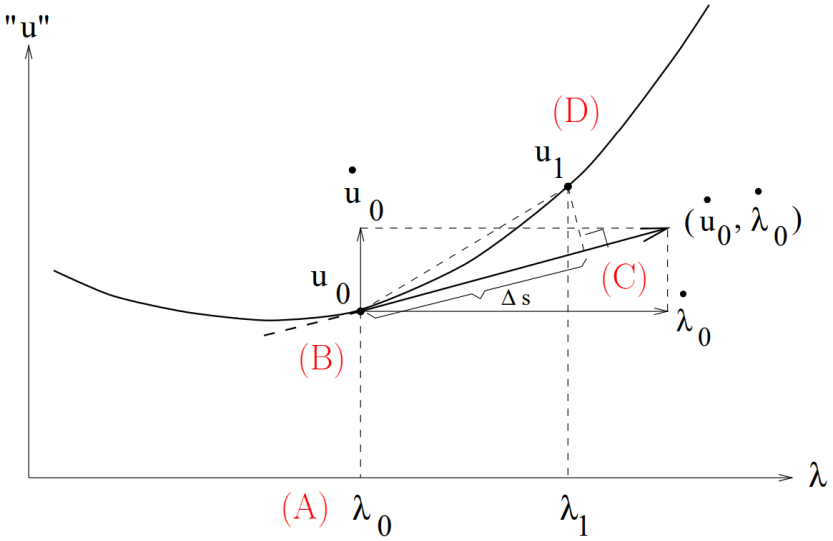

Figure 1.3: Pseudo-arclength continuation process: (A) Choose a starting point on the bifurcation curve that you wish to follow for a parameter value λ_0, with the aim of determining the next point (λ_1, u_1). (B) Evaluate the derivatives of the equations at u_0 with respect to the control parameter and construct a Jacobian matrix and choose a step value Δs, usually called a "pseudo-arclength step", which will determine the step length along the bifurcation curve. Calculate the pseudo-arclength vector $(\dot{u_0}, \dot{\lambda_0})$ by multiplying the tangent vector by the pseudo-arclength step, then update the control parameter by adding the pseudo-arclength step to the current control parameter. (C) Re-evaluate the continuity equations using the new control parameter. (D) Use a numerical method, such as the Newton-Raphson method, to solve the updated continuity equations and find the new solution u_1 of the system. If the solution converges, use it as the point on the bifurcation curve. Repeat the steps until you reach a desired point on the bifurcation curve or decide to stop. (Adapted from [40], slide 48)

Bifurcations allow to obtain a theoretical parallel to Hodgkin's excitability classification (as described in section 1.2.1): in fact, the characteristics of classes 1 and 2 are obtained in the models from the bifurcation structure which passes from the steady state at the periodic regime when the current is varied [34]. Thus, we can characterize the bifurcation diagram of the classes of neurons as follows:

1) The **class I neurons** bifurcation diagrams are characterized by the presence of a *saddle-node on invariant circle* (SNIC). Before the bifurcation, the system has two stationary points, a

[1] Available at https://sites.pitt.edu/ phase/bard/bardware/xpp/xpp.htmll

node point (stable) and a saddle point (unstable). At the bifurcation, the two points merge into a saddle-node point, and an invariant circle appear, hence the name SNIC. This SNIC point is a homoclinic connexion which dissapear, after the bifurcation, to give way to the periodic regime (See Fig. 1.4.A).

2) The **class II neurons** bifurcation diagrams are characterized by the presence of a *Hopf bifurcation* (HB). There are two types of Hopf point: *supercritical* bifurcation and *subcritical*. In the supercritical case, before the bifurcation, we have a stable stationary point. At the bifurcation, a stable periodic point branch is born, and the stability of the stationary point changes to become unstable (See Fig. 1.4.B). In the subcritical case, an unstable stationary point becomes stable at the bifurcation, and the periodic branch which is born is unstable.

3) The **class III neurons** bifurcation diagrams do not have any particular bifurcation point.

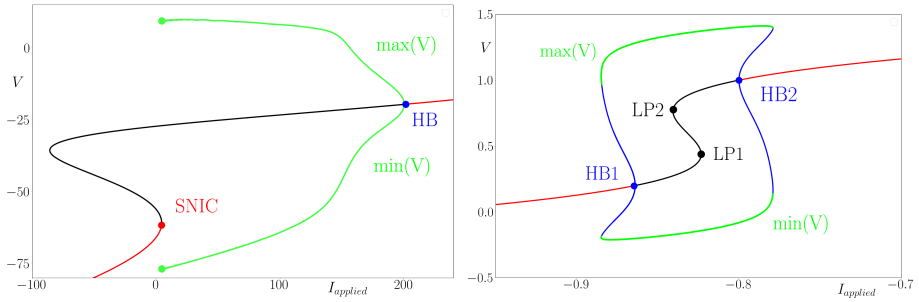

Figure 1.4: Example of bifurcation diagram for (A) a class 1 excitability neuron (Izhikevitch's I_{Na}/I_K model [34]), (B) and a class 2 excitability neuron (FHN model [4]). Red (resp. black) segments of the *S*-shaped curve of equilibria denote stable (resp. unstable) branches. For the example (A), as the applied current I is increased, oscillations (spikes) appear through a SNIC bifurcation (red dot) and then disappear through a supercritical Hopf bifurcation (blue dot). In the case of (B), as the applied current $I_{applied}$ is increased, oscillations (spikes) appear through a first subcritical Hopf bifurcation and then disappear through a second subcritical Hopf bifurcation (blue dots).

Moreover, as evident on Fig. 1.1, a spike is a very nonlinear oscillation, with a fast component (sharp rise and decay of the membrane potential) and a slow component (subthreshold dynamics near the resting state). In models of AP generation, this is closely related to the presence of multiple timescales. In Fig. 1.5, this phenomena is illustrated throught an example: we impose on a class 1 neuron model a slow ramp of current in order to observe the different model regimes, and we overlay the solution on the model bifurcation diagram. Indeed one can observe in Fig. 1.5 the transitions between the regimes (1), (2) and (3). The action potentials occuring in (2) are fast events due to the periodic orbits. Being in the resting state (1) or imposing a too high-level of current until saturation (3) give a slow solution evolving near the critical manifold.

The evolution of membrane potentiel is generally faster than that of variables related to ionic channels. Therefore, one usually studies such models using *slow-fast dynamical systems theory*. This difference in speed between the two phenomena is important and can be highlighted with a methodology called *slow-fast dynamics*. This allows us to study separately slow and fast components of the system, and will be described more fully in the chapter 2.

1.2.3 Integrator and resonator neurons

Excitable systems can be also classified according to the existence of sub-threshold oscillations. This feature divides all systems into two types of neurons: integrator or resonator. Integrator neurons are defined by: a) an ability to excite under high frequency pulses, b) the existence of a precise threshold, and c) the fact that they do not have a sub-threshold oscillation. According to Hodgkin's excitability classification, they are therefore class I excitability neurons (which means

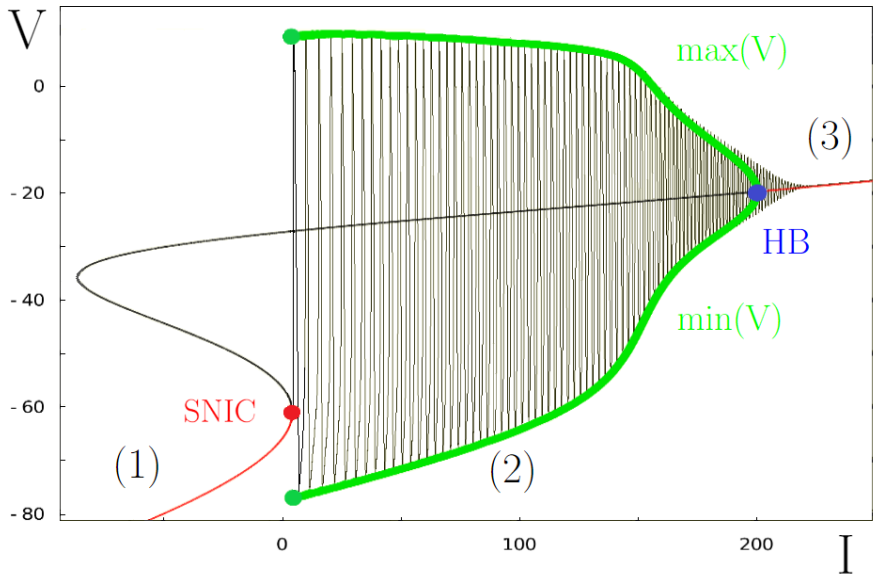

Figure 1.5: Example of bifurcation diagram for a class 1 excitability neuron (Izhikevitch's I_{Na}/I_K model [34]), overlaid with a trajectory coming from a slow current ramp, such as: $\dot{I}_{\text{applied}} = \varepsilon$ where $0 < \varepsilon \leq 1$. (1) The solution follows slowly the critical manifold stable part, (2) then at the SNIC bifurcation, the solution jumps quickly to the periodic regime, and (3) return to a slow regime after the Hopf bifurcation (HB) because of the saturation.

that their frequency tends to 0 during the transition between the stationary and periodic regime). On the contrary, resonator are neurons that: a) respond only to pulses in well-defined frequency intervals, b) do not have a specific threshold, and c) have sub-threshold oscillations. They therefore belong to class II (which means that their frequency does not tend towards 0 during the transition between the steady and periodic regime). In a physiological framework, neurons behave either as integrators or as resonators, and not at the same time.

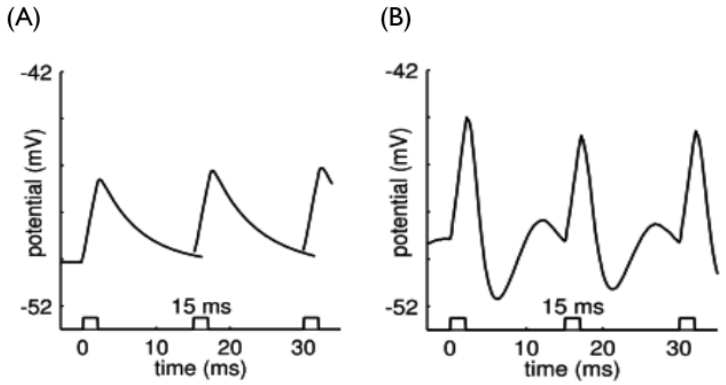

Figure 1.6: Responses sketch of (A) integrators and (B) resonators to input pulses having regular inter-pulse periods. (Adapted from [34], Fig. 7.18)

In Fig. 1.6 (A), the integrator neuron is integrating the signal and going back to the resting state without doing sub-threshold oscillation. On another hand, the resonator neuron (Fig. 1.6 (B)) is also integrating the message, but the potential is oscillating during the transition between the periodic regime and the resting state.

Examples of integrating neurons that can be found *in vivo* include pyramidal neurons in the cerebral cortex, which integrate sensory and motor information to generate movements; Purkinje

neurons in the cerebellum, which integrate sensory information to coordinate fine motor skills; spinal ganglion neurons, which integrate sensory information to control spinal cord reflexes. It is also possible to do the same with resonant-type neurons: the neurons of the cochlear nucleus in the inner ear, which resonate at specific frequencies to help with auditory perception; the neurons of the retina, which resonate at specific frequencies to help detect contours and movements in the visual environment, or neurons in the brain that are involved in memory and cognition, which can resonate at specific frequencies to aid in memory consolidation and thinking.

In the scientific litterature, one can find examples of neurons which are switching from integrator to resonator behavior, but through different methods. In *in vitro* experiments, it is currently possible to make an integrator-type neuron behave like a resonator neuron by means of pharmacological intervention. Indeed, the notion of behavioural switch between integrator and resonator neuron has long been described in the experimental and computational neuroscience literature, however using different approaches. At the experimental level, the environment of the neuron can be controlled in order to obtain this change of excitable behaviour. In particular, this has been achieved pharmacologically, to control the opening of ion channels [31, 41], by current injection [42], using an electric field [43] or even by means of an excitation laser in neuromorphic experiment [44].

In mathematical and computational studies, these methods have also been demonstrated as viable [31, 41, 43–47], together with other approaches: to name a few, by adding terms taking into account new neuronal structures [48, 49], by varying some of the model's parameters [50, 51], or by varying the input forcing frequency [42, 52]. However, all these methods have in common that they result in changes of the system's bifurcation structure in order to allow this transition from integrator to resonator behaviour. In the present case, we want to keep the same bifurcation structure, that is, a SNIC bifurcation associated with integrator-type behaviour, and act differently upon the system so that it can be made to display the characteristic subthreshold oscillations of a resonator.

In chapter 2, we will demonstrate mathematically that an integrator neuron can have the behavior of a resonator neuron once periodically forced in an adequate way, and therefore to have subthreshold oscillations. To do this, we will study the I_{Na}/I_K biophysical model, proposed by E. Izhikevich, in the integrator regime, and apply to it a slow oscillating current, which amounts to adding two slow variables. This addition of two slow variables transforms the initial model into a parabolic bursting model, and it is known that this type of model possesses a "folded-saddle" singularity, then we adjust the forcing in order to make small subthreshold oscillations near this folded singularity, an effect recently unveiled [53]. Effectively, this means that the forced neuron now behaves like a resonator.

1.2.4 Immature neurons

An immature neuron is a neuron that has not yet achieved the differentiation into a mature form. During neuronal development, immature neurons undergo many transformations to reach their mature state. For example, they must make synaptic connections with other neurons to form functional neural networks, and they must also develop dendrites and axons to facilitate the transmission of electrical and chemical signals. Immature neurons play an important role in shaping the developing brain, as they have the ability to divide and differentiate into different types of specialized neurons. This process is called *neurogenesis*, and it is crucial for the formation of functional neural circuits. However, certain types of immature neurons may also be present in adults, such as neural stem cells in certain regions of the brain. These cells have the ability to divide and differentiate into different types of neurons and glial cells, which can help repair brain damage or regenerate damaged tissue.

The excitability of these neurons can vary depending on their stage of development and their location in the brain. In general, they have higher excitability than mature neurons due to the presence of immature ion channels in their cell membrane. These immature ion channels may allow increased sodium entry into immature neurons, making their membrane potential more positive and facilitating the generation of AP, the electrical impulses used to transmit information through the nervous system. Over time, immature neurons undergo morphological and functional changes to reach their mature state. Mature ion channels replace immature channels, reducing sodium entry and increasing the efficiency of neural circuits.

In chapter 3, we will focus on the granule cells present in the dendate gyrus and more precisely on a particular behavior which has been highlighted by colleagues: in fact, during the development of these immature granule cells, the neurons seem to spike for imposed current ranges for which mature neurons will just be saturated. Work on this discovery will therefore focus on the development of a biophysical model reproducing the same phenomenon, and the study of the bifurcation diagrams of this model.

1.3 Excitability in an experimental context

1.3.1 Control Based Continuation in Experiments (CBCE)

As part of this thesis, we want to apply the continuation method in the context of experiments, which means that we do not have access to the equations, but only to the outputs of the experiment, namely, *patch-clamp* electrophysiological recording of neurons. Our approach is motivated by previous work that succeeded in computing bifurcation diagram from noisy experiment, however solely for mechanical experiments [54]. Hence, we aim to adapt this approach to excitable systems, in particular in the present context, neurons.

The use of such methods to obtain a bifurcation diagram is very useful in the field of mathematical models in order to understand the dynamics of a system from the value of a parameter studied. This representation is therefore very important in order to study the transitions between the different regimes (resting state or spikes), but also in order to observe the model excitability threshold. Applying such a method to an experiment would therefore allow us to understand how the behavior of a neuron, whose we do not know underlying dynamics, evolves when we impose a different level of current on it, or even to see its excitability threshold. We now propose to summarize the different methods already existing on this topic.

In [54], they study the motion of a forced pendulum which is described by a nonlinear differential equation, which is difficult to solve analytically. They studied how the solutions of the simple pendulum vary as a function of the forcing amplitude p and they produced bifurcation diagrams (See Fig. 1.7). Moreover, this method is robust against noise which is positive since experiments with neurons are very noisy.

The continuation method applied to the forced pendulum experiment in [54] is based on feedback-control theory. Control theory refers to the idea that systems can be regulated and kept in balance by feedback mechanisms that adjust internal processes in response to an external signal. In a closed-loop system, there is continuous feedback between the output and the input, which allows the control to be adjusted according to system variations. In the chapter 4 and 5, we will explore different strategies to control an experiment (neuron or mathematical model) in such a way that we can apply the continuation process only based on the experiment output. This method is called Control Based Continuation in Experiments (CBCE). As we saw in the previous section 1.2.2, the pseudo-arclength continuation implies that we have access to the differential equations of the model. When we work with experiments, it is of course impossible to access to this kind of information, and so numerical continuation methods cannot be used. We will present in this thesis two continuation methods, based on closed-loop control theories, which can be applied on experiments.

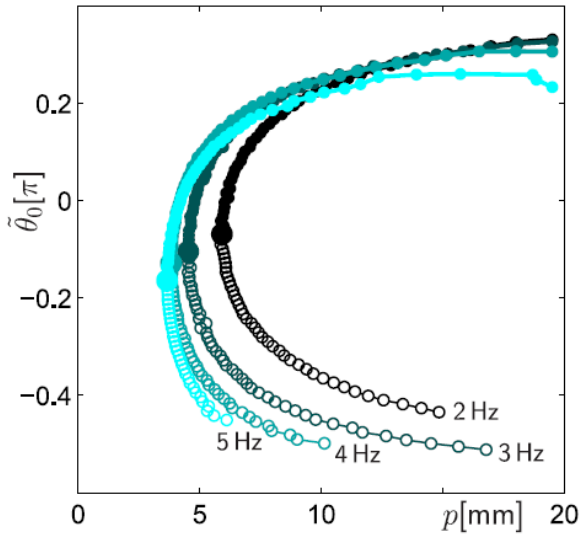

Figure 1.7: Experimental bifurcation diagrams for 2, 3, 4, and 5 Hz, respectively. The variable $\widetilde{\theta}$ is the periodic reference signal, and p is the forcing amplitude. They are showing measured rotations (small circles: hollow for saddle rotations, full for stable rotations) and estimated fold points (large full circles). (Adapted from [54], Fig. 2)

1.3.2 Simulink: from equations to first simulated experiments

Simulink is a graphical simulation and modeling software developed by MathWorks. It is used in many fields of engineering, including computational neuroscience, to simulate and analyze dynamical systems. Simulink's real-time operation enables simulation of real-time systems by using real-time algorithms to control simulation processes. This allows systems and controllers to be tested in a virtual environment before implementing them in the real world. Benefits of using Simulink include the ability to easily visualize systems, add and edit blocks to refine models, debug models in real time, and work with easy-to-use graphical design tools.

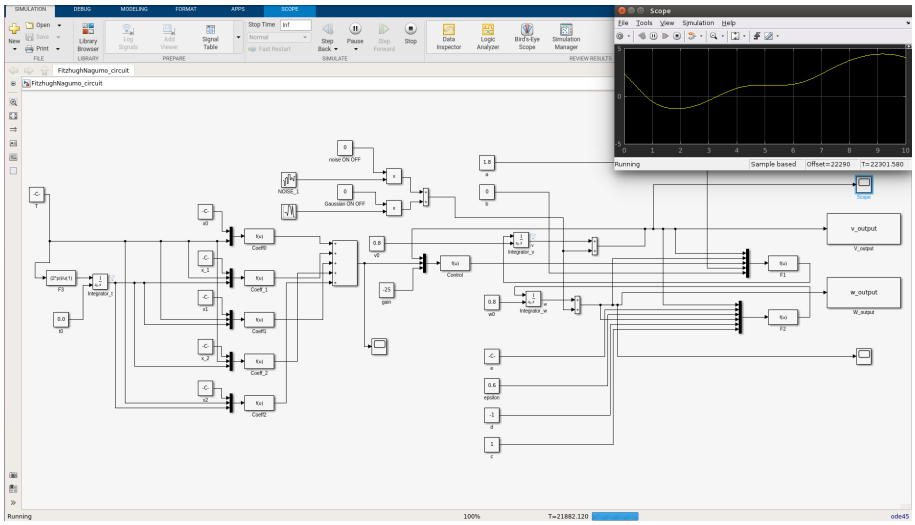

Figure 1.8: Screenshot of the Simulink interface: in the center of the image, we can see the FHN model modeled in the form of a circuit. Each component corresponds to an operation or term of the model. In the upper right corner, the scope gives the dynamics of the circuit in real time, which makes it possible to follow the behavior of the model.

The main advantage of this Matlab toolbox is the fact that we know exactly the equations that

govern the dynamics of the experiment, but also the dynamics themselves, which gives us points of reference for testing the CBCE algorithm that we wish to implement. In addition, it is possible to disable/enable the noise, which gives us even more control over the experience. This step is therefore an interesting testbed for CBCE.

In computational neuroscience, Simulink is used to model neural systems, in particular to study the dynamics of neural networks and to understand information processing processes in the brain. Applications of Simulink in computational neuroscience include modeling vision, modeling synaptic plasticity, simulating auditory perception, and studying the interactions between different levels of information processing in the brain.

1.3.3 Patch-Clamp: application to neurons

Modern electrophysiology has emerged through the pioneering work of a few key scientists. George Marmont, who designed the *space clamp* protocol [55]; Kenneth Cole, who with Marmont and others, adapted the previous protocol to the *voltage clamp* [56]. And, at the same time and in close contact with Marmont and Cole, Alan Hodgkin, Andrew Huxley and Bernard Katz, who exploited the voltage clamp to measure conductances [1–3]. Hodgkin and Huxley were awarded the Nobel Prize (1963 Nobel Prize in Physiology or Medicine) in part for these voltage-clamp measurements. Much later on, in the early 1980s, the modern *patch-clamp* technique was invented [57, 58], for which Neher and Sackmann were awarded the Nobel Prize (1991 Nobel Prize in Physiology or Medicine).

An even more recent and more advanced electrophysiology protocol, namely *dynamic clamp*, was invented in the early 1990s [59, 60] and it allows a two-way real-time communication between the experimental neuron and a computer simulation. For instance, it allows to replace a pharmacologically blocked ion channel by a computer simulation of it and study the response of the controlled cell to variations of all parameters of this model, hence validating the model and exploring the space of cellular responses. This technique has become very popular in electrophysiology labs worldwide and it can be used to bring experiments on neurons closer to their mathematical and computational description; see e.g. [61–63]. For instance, in [64] it was used to build directly from controlled neural experiments phase diagrams, namely a set of states characterising a large number of possible excitable states of the neuron. These bring unique information about how neurons adapt to variations of external of internal inputs (e.g. conductances, concentrations).

Specifically, patch-clamp is a technique used to study the properties of ion channels in living cells. The technique consists of using a glass micro-pipette filled with an ionic solution of defined composition to establish electrical continuity with the membrane of an isolated living cell. The technique can be used to study excitable cells such as neurons and muscle cells as well as non-excitable cells which also have ion channels on their surface. The patch-clamp technique can be used in clamped voltage mode (voltage-clamp), which makes it possible to measure the membrane current at a potential determined by the experimenter, or in clamped current mode (current-clamp), making it possible to to inject charges while measuring the membrane potential.

1.3.4 Recurrence Structure Analysis: A way to detect metastable states

In the final chapter of this thesis, we will focus on a set of data from mice who have been presented with different types of chemicals and whose olfactory bulb activity is monitored using *calcium imaging*. The main aim is to determine whether certain groups of neurons are more active than others in recognising an odour type in different individuals: to do this, a pipeline was developed to map the olfactory bulbs of the three subjects, which allowed the neurons to be compared. A method was then used that combined several data processing techniques: first, the K-means method was used to group the data from each mouse into regions of interest (also called clusters) [65–67]. This allowed the identification of populations of neurons that were more active in

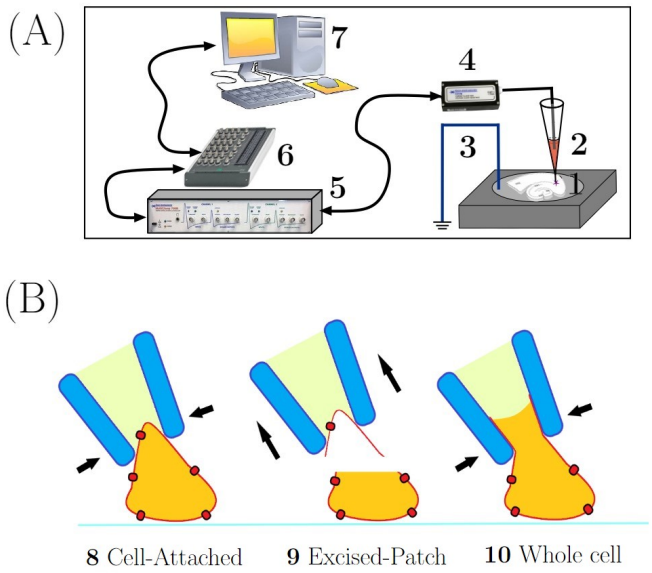

Figure 1.9: (A) Experimental setup with brain slice (**1**), patch pipette (**2**), reference electrode (Ag-AgCl pellet) connected to ground (**3**), amplifier ((**5**), Multi-clamp 700B) with CV-7B headstage (**4**), AD-converter ((**6**), National Instruments NI USB-6343) and standard PC computer (**7**). (B) Sketch of the Patch-clamp main steps: the neuron is fixed to a solid support and a glass micropipette electrode is used for the recording. Then, various recording configuration are possible: (**8**) the micropipette is placed on the cell, and it aspires this surface for the records, (**9**) the micropipette is recording only an excised patch of the neuron; depending of the sense of the patch, the method is called "outside-out" or "inside-out", (**10**) the cell is open to the environment of the micropipette for the records.

response to certain chemicals. A data processing technique was then used to reduce the noise in the recordings [68].

The *Recurrence Structure Analysis* (RSA) method [22–25, 69, 70] was then used to generate diagrams, called *recurrence plots* (RP), showing when a phase space trajectory visits roughly the same region in the phase space. Recurrence is a fundamental property of many dynamical systems (i.e. systems that evolve in time), which was first formulated by Poincaré [20] and is ubiquitous of various processes in nature and biological systems. A system may strongly diverge, but after some time it recurs infinitely many times as close as one wishes to its initial state. In other words, certain regions of state space (of the associated system) are frequently visited in the course of time. A visualisation of this recurrence was first introduced by Eckmann *et al.* [21] and different measures can be used to study the properties of the system. However, a fundamental threshold parameter, ε, is used to identify and determine the level of observation of the recurrence. To provide an optimal observation, it is possible to optimize the value of ε, which allows to more precisely dissect the transient and metastable states (i.e. dynamical quasi-invariant states) of a system [22–25]. The applicability to brain dynamics has been demonstrated. [24]. Specifically, the RSA method identifies an optimal condition (utility function parameterised by ε) based upon a stochastic matrix. The utility function is constructed by the trace of the stochastic matrix and the entropy of the columns and rows of the matrix, and finally it is maximised by an optimisation method, which determines the optimal ε. This then enables to unveil the recurrent states (given as transient quasi-invariant states or metastable states) of a given system.

The metastable states are understood as saddle attractors (i.e., attractors with one stable manifold from which system's trajectories approach the attractor and one unstable manifold via which system's trajectories are repelled) in the phase space of a system. For example in our present case, then the phase space is the combined the tufted cell activity, which is high-dimensional, and

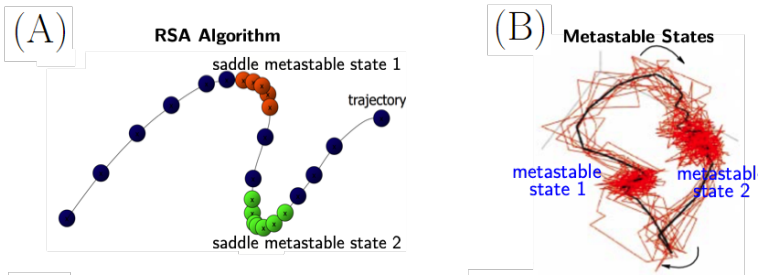

Figure 1.10: RSA illustrating sketch: (A) Representation of a trajectory in some phase space, where balls represent the discrete times at which the signal is measured. Near the saddle metastable states, the balls are overlapping due to the slowing down of the trajectory: these overlapping balls are the metastable states. (B) Representation of the same concept but with a noisy signal. Here, two metastable states are also founded.

of which we have reduced the dimensionality with the K-means method. The determination of saddle attractor is possible because the trajectory slows down in the vicinity of a saddle, and this will be reflected by longer time scales in the recurrence domains. This idea is shown in the sketch Fig. 1.10 (A), which depicts a curve as a system's trajectory in some phase space and the balls represent the discrete times at which the signal is measured. Note that these balls will overlap close to the saddle attractor due to slowing down of the trajectories. Subsequently, a colouring scheme (i.e. symbols which are then associated with the symbolic dynamics) are used to distinguish between the transient states and saddles. The blue balls are the transient states, while the two identified saddles are orange and green, respectively, in this case (i.e. in this cartoon we have two identified saddles). In Fig. 1.10 (B) we represent the same idea of a noisy trajectory which recurs between two metastable states. Once the transient and metastable states are identified, then the associated Markov model can be constructed to explain an experimental observation [24].

1.4 Thesis organization

In Chapter 2, we propose a new strategy to obtain an excitability switch in a neuron model, based upon the presence of multiple timescales in this model. We illustrate our approach with the so-called I_{Na}/I_K model by E.M. Izhikevich, studied under the effect of a slow sinusoidal external current. This creates a slow-fast structure characterized by the presence in phase space of a special point called *folded-saddle singularity*. Using recent results on geometric singular perturbation theory related to folded saddles, we modify the slow forcing in order to obtain algebraic conditions on the folded saddle which are known to create small-amplitude oscillations in trajectories flowing past it in phase space. In the neuronal context, we show that these small-amplitude oscillations are subthreshold and hence endow the integrator model with the characteristics of a resonator. This effect is not dependent upon the choice of type-I model, it only requires the associated bifurcation structure (SNIC) of a type-I model. We also show a scenario for the reverse switch, from resonator to integrator, again based on multiple-timescale dynamics but this time reliant upon a *folded-node singularity*. Folded nodes are known to be associated with small-amplitude oscillations which, in the neuronal context, are subthreshold.

Chapter 3 was concerned with the analysis of neuronal data obtained from *patch-clamp* recordings of immature Granule cells (GC) during their development. The excitability of these neurons can vary depending on their stage of development and their location in the brain. In general, they have higher excitability than mature neurons due to the presence of immature ion channels in their cell membrane. These immature ion channels may allow increased sodium entry into immature neurons, making their membrane potential more positive and facilitating the generation of AP, the

electrical impulses used to transmit information through the nervous system. Over time, immature neurons undergo morphological and functional changes to reach their mature state. Mature ion channels replace immature channels, reducing sodium entry and increasing the efficiency of neural circuits.

In Chapters 4 and 5, we focused on obtaining bifurcation diagrams from noisy experiments and numerical continuation methods [17, 18]. Based upon closed-loop feedback control, with an embedded rootfinder algorithm (e.g. Newton's method), the Control Based Continuation on Experiments (CBCE) method allows to overcome the fact that we do not a priori know the governing equations associated with the experiment we are studying. In chapter 4, we studied experimental data obtained by subjecting a neuron to two different patch-clamp electrophysiology protocols: a voltage clamp with slow ramp on the hold voltage (VC), and a current-clamp with slow ramp on the hold current (CC). The result of the VC protocol provided a (current, voltage) curve resembling a steady-state bifurcation diagram. What is more, once superimposed onto the result of the CC protocol, the two datasets interacted in a similar manner as a full systems solution overlaid onto a fast-subsystem steady-state bifurcation diagram. That is, a slow-fast dissection effect. In this chapter, we explained why this simplified method works, and we pose hypotheses on the question of the noise obtained around the unstable parts of the critical manifold. Thus, this method, which is akin to a simplified revision of CBCE (without the Newton part) allows us to obtain a viable approximation of an experimental steady-state bifurcation diagram, and also it can validate the underlying neuron model as a good representation of the real neuron.

In Chapter 5, we focused on applying a complete version of the CBCE method on experiments simulated via Simulink. Simulink is a graphical simulation and modeling software developed by MathWorks. It is used in many fields of engineering, including computational neuroscience, to simulate and analyze dynamical systems. Simulink's real-time operation enables simulation of real-time systems by using real-time algorithms to control simulation processes. This allows systems and controllers to be tested in a virtual environment before implementing them in the real world. Benefits of using Simulink include the ability to easily visualize systems, add and edit blocks to refine models, debug models in real time, and work with easy-to-use graphical design tools.

In Chapter 6, we presented a data-analytic study on the olfactory bulb of mice. Namely, we analysed calcium imaging data coming from 3 mice subjected to different chemicals. We developed a pipeline to study the data: first we segmented the real coordinate space of the neurons with the k-means method, then we produced the average of the neurons of each segment for each individual. Then, we pre-processed the data with the baseline alignment method, and finally applied the RSA method. These first results showed us that there are indeed patterns of recurrence of these olfactory recognitions. Next, having segmented the data into subgroups based on the coordinates of neurons in the olfactory bulb allows us to compare the centroids of these metastable states. These results will be summarized in odotopic maps which demonstrate that the same areas of the olfactory bulb are activated for the same chemical. In addition, we will also demonstrate that for the different types of chemicals the same areas are used for recognition but at different intensity levels. We will finally propose the learning of a supervised classifier of these centroids to show that they are significantly sufficient to discriminate them.

In Appendix 8, we propose the study of a neuromorphic circuit based on the Morris-Lecar model. A neuromorphic circuit is a type of electronic circuit designed to reproduce the operation of neural networks in the human brain, this includes the simulation of neuronal electrophysiology, such as the generation of action potentials, synaptic transmission, and other processes electrical and chemical linked to neuronal activity. The interest in studying a single neuron neuromorphic circuit for this thesis was to have a templated experiment in which we know the equations which lead the behavior of the model. What we hoped for was to be able to apply our methodologies

developed during the thesis on this circuit in order to calibrate them, but also to work on the notion of real-time experience, which was a new challenge for these methods. In open-loop setup, the experiment worked well, however, with the closed-loop setup we encountered problems that could not be fixed. Namely, delays between the application of the given current to the experiment, and reading the output.

Chapter 2

From integrator to resonator neurons: A multiple-timescale scenario

The results presented in this chapter have been published [7] as: G. Girier[1], M. Desroches[2] and S. Rodrigues[1,3], From integrator to resonator neurons: A multiple-timescale scenario, Nonlinear Dynamics ***111****: 16545-16556, 2023. https://link.springer.com/article/10.1007/s11071-023-08687-1*

Now that the context of this thesis is established, we will begin our exploration of neuronal excitability. As described previously, this chapter will lead us to explore the modulation of the behavior of an integrator-type neuron so that it adopts a resonator-type behavior, while retaining its mathematical properties inherent to an integrator neuron. This information could therefore help us to understand with more perspective the transmission of information *in vitro*, but also the plasticity that a neuron can have when external currents are transmitted to it. This step will allow us to further immerse ourselves in the mathematical and computational aspect of our research, and thus establish the different tools that we will use throughout the thesis to analyze systems of differential equations.

2.1 Introduction

Excitable systems, in particular neurons, can be classified according to the various criteria, one of them being the existence of sub-threshold oscillations [29–31]. This feature allows to distinguish between two types of neurons: *integrator* and *resonator*. Integrator neurons are defined by: a) the ability to get excited under high frequency pulses, b) the existence of a precise threshold, and c) the fact that they do not have a sub-threshold oscillations. They belong to what is usually referred to as type-I neuronal excitability, which means that their firing frequency starts from 0 at the transition between the stationary and the periodic regime. In contrast, resonator are neurons that: a) respond only to input with well-defined frequencies (they "resonate" with these special frequencies), b) do not have a well-defined threshold, and c) have sub-threshold oscillations. Therefore they display type-II neuronal excitability, which means that their firing frequency is bounded away from 0 at the transition between the stationary and the periodic regimes. In an experimental framework, neurons behave either as integrators or as resonators, and this feature is observed neither simultaneously nor in the same physiological conditions.

From a dynamical systems standpoint, the underlying models of these two types of neurons differ by their bifurcation structure upon variation of an applied current I as main parameter. Namely, in integrator-type models a *saddle-node on invariant circle (SNIC)* bifurcation organises the transition from rest to spiking, and their excitability threshold is defined by the stable manifold

[1] BCAM Basque Center for Applied Mathematics, Bilbao, Bizkaia, Spain
[2] MathNeuro, Project-Team, Inria Centre, Université Côte-d'Azur, France
[3] Ikerbasque, the Basque Foundation for Science, Spain

of the saddle equilibrium that disappears through the SNIC bifurcation. In contrast, in resonator-type neuron models this transition occurs via a *Hopf bifurcation* (HB), which is often subcritical and followed, in parameter space, by a saddle-node bifurcation of limit cycles [71]. The threshold is not well-defined however it can be approximated by a family of so-called *canard cycles* [34, 72–74].

In *in vitro* experiments, it is currently possible to make an integrator-type neuron behave like a resonator neuron by means of pharmacological intervention. Indeed, the notion of behavioural switch between integrator and resonator neuron has long been described in the experimental and computational neuroscience literature, however using different approaches. At the experimental level, the environment of the neuron can be controlled in order to obtain this change of excitable behaviour. In particular, this has been achieved pharmacologically, to control the opening of ion channels [31, 41], by current injection [42], using an electric field [43] or even by means of an excitation laser in neuromorphic experiments [44].

In mathematical and computational studies, these methods have also been demonstrated as viable [31, 41, 43–47], together with other approaches: to name a few, by adding terms taking into account new neuronal structures [49], by varying some of the model's parameters [50, 51], or by varying the input forcing frequency [42, 52]. However, all these methods have in common that they result in changes of the system's bifurcation structure in order to allow this transition from integrator to resonator behaviour. In the present case, we want to keep the same bifurcation structure, that is, a SNIC bifurcation associated with integrator-type behaviour, and act differently upon the system so that it can be made to display the characteristic subthreshold oscillations of a resonator.

The main objective of the present work is to demonstrate mathematically that an integrator neuron – namely, a type-I neuron model– can be made to behave like a resonator neuron once an adequate slowly-varying current is applied to it with real-time feedback from the membrane potential; hence, we aim to obtain subthreshold oscillations in an integrator neuron model. Crucially, we want to achieve this apparent excitability switch without modifying the underlying bifurcation structure of the model.

To do so, we will exploit the multiple-timescale structure of the slowly-forced integrator model and show that subthreshold oscillations are possible in a specific parameter range, no matter which integrator model we are starting from, provided it has a SNIC bifurcation upon constant applied current and provided we apply to it a specific slowly-varying time-dependent current.

In a nutshell, we will show that the forcing requires to consider two additional slow variables and that the extended (minimally 4D) model possesses a so-called *folded-saddle singularity* [26, 27]. It was recently discovered [53] that subthreshold oscillations can appear near a folded saddle provided a certain algebraic condition is satisfied in the singular limit, that is, when the (explicit) timescale separation parameter ε tends to 0; see also [28]. It turns out that this condition cannot be obtained in a slowly-forced integrator system if the forcing is too simple, that is, harmonic; see Section 2.3. As we will show in Section 2.4, one needs a feedback term from the voltage in the forcing equation in order to obtain the subthreshold oscillations, which suggests in the context of real neurons, an *autaptic* behaviour, and can be tested experimentally using a *dynamic-clamp* protocol.

We will showcase our strategy with a simple biophysical example of 2D type-I neuron model, namely the $I_{\mathrm{Na}}/I_{\mathrm{K}}$ model proposed by Izhikevich in [34]; however our approach will work with any type-I neuron model. Noteworthy, it does not require to alter the underlying bifurcation structure of the model, which is customary in studies reporting a switch from integrator to resonator [31, 43, 47]. The excitability switch that we propose here is purely due to timescale separation between the model and the forcing. We also showcase the reverse scenario, namely a switch from resonator to integrator. This requires to have a folded node instead of a folded saddle, therefore a different slow forcing structure, however here again we obtain the switch from one neuronal type to the other by staying within the same bifurcation scenario, only playing with the slow-fast structure of the model

and of its folded singularity. The folded-node scenario indeed induces a particular geometry for the trajectories passing near such a folded singularity. Namely, they make transient small-amplitude oscillations, which correspond to subthreshold oscillations in the neuronal context [26, 74] and their number can be controlled, e.g., by varying initial conditions. Indeed, families of initial conditions giving rise to the same number of oscillations form so-called *rotation sectors* in phase space. Furthermore, by controlling the trajectory to flow into the first rotation sector, one can suppress these subthreshold oscillations and hence, turn the behaviour into an integrator, hence obtaining the reverse switch. However, we will argue that the folded-saddle scenario is more appropriate for this switch between integrator and resonator neuron in order to obtain a behaviour as close as possible to experiments.

This chapter is organised as follows. First, in section 2.2, we will provide all the theories needed to understand properly the results obtained. In Section 2.3, we present the I_{Na}/I_K model and analyse numerically its integrator structure. Then, we apply to it a first slow harmonic forcing and show that it is sufficient to create a folded-saddle singularity but insufficient to obtain subthreshold oscillations, which require a more elaborate forcing. This is why, Section 2.4, we adapt the forced current in order to obtain the singular-limit algebraic condition giving rise to subthreshold oscillations in the full model, and hence the resonator behaviour. In Section 2.5, we present the reverse scenario, whereby a resonator neuron can behave like an integrator, and show that this is due to another type of folded singularity, namely a *folded-node singularity*. Finally, we conclude in Section 2.6 and propose a strategy to verify experimentally our theoretical predictions.

2.2 Theoretical context

In this section, the different theories and methods necessary for the understanding of the chapter will be given.

2.2.1 Slow-fast dynamics

Definition: Suppose one have a system with two timescales (t and τ being the characteristic time scale for the process):

$$\begin{cases} \tau_x \dot{x} &= f(x,y), \\ \tau_y \dot{y} &= g(x,y), \end{cases} \tag{2.1}$$

where $(x,y) \in \mathbb{R}^n \times \mathbb{R}^m$, τ_x (resp. τ_y) is the characteristic timescale of x (resp. y), and the overdot denotes differentiation with respect to the time t.

Suppose x is the variable evolving on fast time scale t. Suppose y is the variable evolving on slow time scale τ. Suppose there is a ratio between this time scale t and τ. Then suppose that x evolves on a much faster timescale than y, which can be expressed as: $0 < \tau_x \ll \tau_y$. Then one can define a *timescale ratio parameter* ε by: $\varepsilon := \tau_x/\tau_y$, which immediately gives that $0 < \varepsilon \ll 1$, and system (2.1) can be rewritten in explicit slow-fast form as:

$$\begin{cases} \varepsilon \dot{x} &= f(x,y), \\ \dot{y} &= g(x,y), \end{cases} \tag{2.2}$$

where t naturally appears as *slow time* in (2.2). Only then, one can introduce the *fast time* τ by posing: $\tau = t/\varepsilon$, which gives:

$$\varepsilon d\tau = dt \tag{2.3}$$

Using chain rule, we obtain:

$$\frac{dx}{d\tau} = \frac{dx}{dt}\frac{dt}{d\tau} = \varepsilon\frac{dx}{dt} \tag{2.4}$$

We rewrite 2.1 in terms of fast time τ:

$$\begin{cases} x' &= f(x,y), \\ y' &= \varepsilon g(x,y). \end{cases} \tag{2.5}$$

The two systems of differential equations (2.2) and (2.5) have the same phase portraits, as long as $\varepsilon \neq 0$, but their trajectories are parameterized differently. The interest of these two systems (2.2) and (2.5) lies in the fact that their limits when $\varepsilon = 0$ are very different. These two limits allow to highlight separately fast and slow components, respectively, of the overall system dynamics.

For system (2.2), the singular limit $\varepsilon = 0$ yields:

$$\begin{cases} 0 &= f(x,y) \\ y' &= g(x,y) \end{cases} \tag{2.6}$$

System (2.6) is a *differential-algebraic equation* (DAE) usually referred to as *slow subsystem* (sometimes also called *reduced problem*). Specifically, this is a differential equation for the slow variables y together with an algebraic constraint given by $f(x,y) = 0$, which effectively defines the phase space of the system and which corresponds to the equation of the so-called *critical manifold* (fast nullmanifold). So, this system allows us to study the limiting slow dynamics of the system (slow flow).

For system (2.5), the singular limit takes the form:

$$\begin{cases} x' &= f(x,y,0) \\ y' &= 0 \end{cases} \tag{2.7}$$

System (2.7) is typically called *fast subsystem* or *layer problem*, it is a form of *adiabatic* limit. In the fast subsystem, the slow variables y are kept constant and hence become parameters in the fast equations, which persist. The set of stationary points of the fast subsystem with respect to the parameters y is therefore defined by the set $\{f(x,y,0) = 0\}$, and we see that we recover the equation of the critical manifold. Thus, the critical manifold plays an essential role both in the fast subsystem, as a geometric representation of the family of stationary points w.r.t y, and in the slow subsystem, as phase space.

Thanks to slow-fast systems for a precise range of parameters, *canard solutions* can appear. Canard solution are particular trajectories, and a strict minimum system, with one fast and one slow equations, is enough to obtain them. The best known class is the case where the critical manifold of the studied system forms a regular fold, and have an attracting and a repellant part.

The behavior of these solutions is the next: first, they follow the attracting part of the critical manifold, passes close to a bifurcation point, and then follow the repelling part of critical manifold during a short delay, before to be ejected. This interesting phenomenona is also due to the loss of normal hyperbolicity at the bifurcation point. (See Fig. 2.3 for an example of canard solution)

In geometric terms a canard solution corresponds to the intersection of an attracting and a repelling slow manifold near a non-hyperbolic point of S. We call the intersection of these fixed slow manifolds a maximal canard. If a slow-fast system with canard solutions has a dimension superior or equal to 3, it is so possible to apply the desingularisation and reduction process, in order to highlight particular folded-fixed points which explain the particular behavior of these solutions.

Another important concept is the slow passage phenomena which is the results of a slow variation of a parameter. The consequence of the slow passage effect is that bifurcation does not occur at the bifurcation point, but, further, with a delay after the passage throught the bifurcation point in question. These dynamics can be observed around bifurcation points like Hopf bifurcations, pitchfork bifurcations, or SNIC bifurcations.

The prediction of the model behavior can be difficult to evaluate because of this bifurcation-delay. It becomes even worse due to the presence of jump transitions and memory effect associated with the slow passage effect. So, delay control in bifurcation due to the slow passage effect is something to take in consideration in order to understand model mecanisms. This phenomena could be compared to a canard phenomena, which is also a delay-bifurcation, but this new one is non-explosive and it is not restricted to a very narrow parameter intervals.

2.2.2 Desingularized Reduced System

Definition: A study of new particular dynamics of solutions which progress near new points of equilibrium, the pseudo-stationary points, will also interest us. These new concepts therefore require a better understanding of the structure of the slow subsystem of a slow-fast system of type 2.6, but with a variable x of dimension 1, and y, of dimension 2, which means that we have now two slow variables that we will call y and z:

$$\begin{cases} \varepsilon x' &= f(x,y,z,\varepsilon) \\ y' &= g(x,y,z,\varepsilon) \\ z' &= h(x,y,z,\varepsilon) \end{cases} \tag{2.8}$$

A change of time in 2.8 therefore makes it possible to obtain the fast subsystem:

$$\begin{cases} x' &= f(x,y,z,\varepsilon) \\ y' &= \varepsilon g(x,y,z,\varepsilon) \\ z' &= \varepsilon h(x,y,z,\varepsilon) \end{cases} \tag{2.9}$$

From 2.8, with $\varepsilon = 0$, we derive the following implicit equation with respect to time, thanks to the chain rule:

$$\begin{aligned} & 0 = f(x,y,z,0) \\ \iff\ & 0 = \frac{\partial f}{\partial x}(x,y,z,0)x' + \frac{\partial f}{\partial y}(x,y,z,0)y' + \frac{\partial f}{\partial z}(x,y,z,0)z' \end{aligned} \tag{2.10}$$

By replacing the terms that we know of the complete system, we therefore obtain the following rearrangement:

$$-\frac{\partial f}{\partial x}(x,y,z,0)x' = \frac{\partial f}{\partial y}(x,y,z,0)g(x,y,z,0) + \frac{\partial f}{\partial z}(x,y,z,0)h(x,y,z,0) \tag{2.11}$$

However, equation 2.11 is no longer defined when $\frac{\partial f}{\partial x} = 0$, i.e. along the fold of the critical manifold: it is therefore necessary to desingularize our system, or to project it on a plane. Indeed, given that we have two slow variables, the critical manifold will be dimension two, and therefore the slow subsystem will be essentially dimension two as well. Thus, one projects in such a way as to keep only two differential equations:

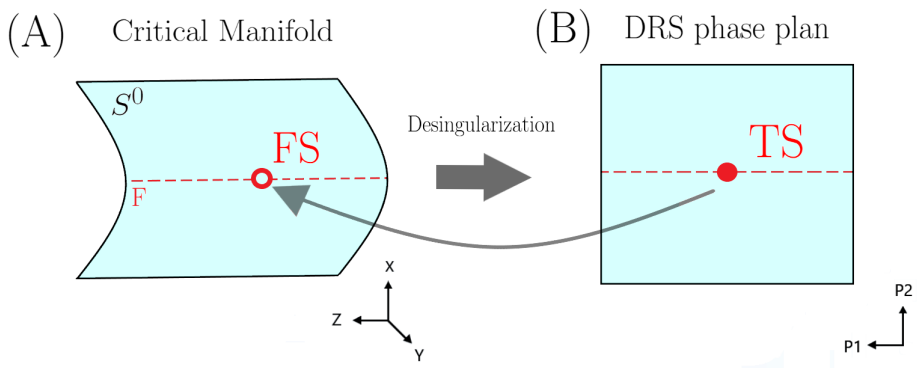

Figure 2.1: Diagram of the principle of desingularization of a system, with (A) a critical manifold which has a fold where the solutions passing through it are not defined, which gives (B) a desingularized reduced system whose phase plane can be studied in order to to emerge a true singularity (TS) present on what is the fold of the critical manifold. Then, we recover the flow of the slow subsystem on the critical manifold (A) where the folded-singularity (FS) is. Also shown is the critical manifold S^0 (blue surface), and the fold curve F (dotted line).

$$\begin{cases} x' &= \frac{\frac{\partial f}{\partial y}(x,y(x,z),z,0)g(x,y(x,z),z,0)+\frac{\partial f}{\partial z}(x,y(x,z),z,0)h(x,y(x,z),z,0)}{\frac{\partial f}{\partial x}(x,y(x,z),z,0)} \\ z' &= h(x,y(x,z),z,0) \end{cases} \tag{2.12}$$

This system also not being defined when $\frac{\partial f}{\partial x} = 0$, we therefore admit an unusual change in time depending on x, and which makes it possible to obtain the following auxiliary system:

$$\begin{cases} x' &= \frac{\partial f}{\partial y}(x,y(x,z),z,0)g(x,y(x,z),z,0) + \frac{\partial f}{\partial z}(x,y(x,z),z,0)h(x,y(x,z),z,0) \\ z' &= -\frac{\partial f}{\partial x}(x,y(x,z),z,0)h(x,y(x,z),z,0) \end{cases} \tag{2.13}$$

Where $y(x,z)$ therefore only depends on x and z since we place ourselves on the critical manifold and this dependence amounts to locally solving the equation of the critical manifold: $f(x,y,z,0) = 0$.

The system of equation 2.13 corresponds to what is called the desingularized reduced system (DRS): it makes it possible to obtain stationary points in its phase plane which will correspond to folded singularity points in the phase plane of the complete system studied. Thus, from 2.13, we can linearize around such a stationary point and we obtain the following Jacobian matrix:

$$\mathbf{J} = \begin{pmatrix} A(x,z) & B(x,z) \\ C(x,z) & D(x,z) \end{pmatrix} \tag{2.14}$$

$$\begin{aligned} \mathrm{tr}(J) &= A(x,z) + D(x,z) \\ \det(J) &= A(x,z)D(x,z) - C(x,z)B(x,z) \\ \Delta(J) &= Tr(J)^2 - 4\det(J) \end{aligned} \tag{2.15}$$

If $\det(J) < 0$, then the stationary point obtained is a saddle point in the phase plane of (6), and will therefore give a folded-saddle point in the complete system 2.8. If $\det(J) > 0$, $\mathrm{tr}(J) < 0$, and $D(J) > 0$, we obtain an attractive node point (which will give a pseudo-node point in the complete system), whereas if $\det(J) > 0$, $\mathrm{tr}(J) > 0$, and $D(J) > 0$, the node point is repulsive. If

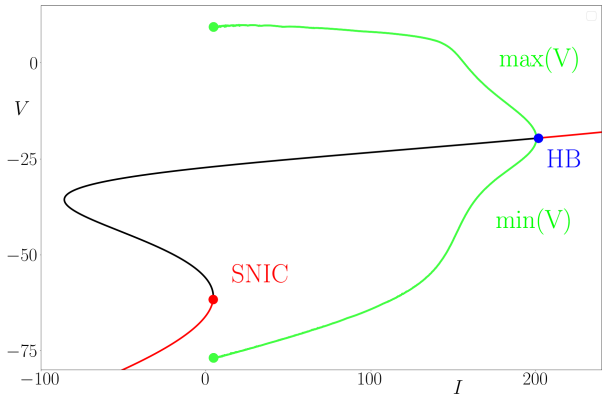

Figure 2.2: Bifurcation diagram of the system (2.16) with respect to parameter I. Red (resp. black) segments of the S-shaped curve of equilibria denote stable (resp. unstable) branches. As the applied current I is increased, oscillations (spikes) appear through a SNIC bifurcation (red dot) and then disappear through a supercritical Hopf bifurcation (labelled HB and marked by a blue dot). Parameter values are: $C=1$, $E_{\mathrm{L}}=-80$, $E_{\mathrm{Na}}=60$, $E_{\mathrm{K}}=-90$, $g_{\mathrm{L}}=8$, $g_{\mathrm{Na}}=20$, $g_{\mathrm{K}}=10$, $V_{m,1/2}=-20$, $K_m=15$, $V_{n,1/2}=-25$, $K_n=5$, $\tau_n(V)=1$.

$\det(J)>0$, $\mathrm{tr}(J)<0$, and $D(J)<0$, we obtain an attractive focus (which will give a folded-focus in the complete system), while $\det(J)>0$, $\mathrm{tr}(J)>0$, and $D(J)<0$, then the focus is repellent.

2.3 Izhikevich's $I_{\mathrm{Na}}/I_{\mathrm{K}}$ model.

2.3.1 Under constant external current

We consider a two-dimensional conductance-based neuron model with minimal components for excitability. This model was proposed by Izhikevich in [34] and it was referred to as the $I_{\mathrm{Na}}/I_{\mathrm{K}}$ model, since it only assumes basic persistent sodium with instantaneous activation, potassium and leak currents; this is the name we shall use throughout this article. The model's equations are:

$$\begin{aligned} CV' &= I - g_{\mathrm{L}}(V-E_{\mathrm{L}}) - g_{Na}m_\infty(V)(V-E_{Na}) - g_K n(V-E_K), \\ n' &= \frac{n_\infty(V)-n}{\tau_n(V)}, \end{aligned} \tag{2.16}$$

with steady-state functions: $x_\infty(V)=(1+\exp\big((V_{x,1/2}-V)/k_x\big))^{-1}$, $x=\{m,n\}$, and where the prime denotes differentiation with respect to the time τ.

For simplicity, we take the time constant $\tau_n(V)$ to be independent of V and we shall fix its value to 1. System (2.16) is based on a simplification of the two-dimensional reduction of the Hodgkin-Huxley model proposed by Krinskii & Kokoz [75], independently by Rinzel [76], and further studied, e.g., by Moehlis [77]. Thus, the variable V represents the membrane potential of the neuron, and n, the activation of potassium channels. The constants g_x (where x corresponds to L, Na, or K) are the maximal conductances of the ionic currents considered, E_x are the Nernst potentials of the ionic species and C is the capacitance of the neural membrane; I denotes an externally applied current.

The bifurcation diagram of system (2.16) with respect to I, shown in Fig. 2.2, is typical of a neuron with type-I excitability [78]. Namely, a family of low-voltage equilibria (rest states of the neuron) destabilise and give way to a family of stable limit cycles (spiking states of the neuron) via a SNIC bifurcation, which occurs at an input current value $I\approx 4.51$. The SNIC bifurcation being a homoclinic-type bifurcation, the emerging stable cycle has a very large period (tending to infinity

at the bifurcation), hence a very small frequency, which is a key hallmark of type-I excitability. At a much higher value of the input current, the branch of stable cycles disappears via a supercritical Hopf bifurcation at $I \approx 200$. Therefore, system (2.16) is considered to be in an integrator regime here.

2.3.2 Applying a slow sinusoidal external current

We now consider a periodic forcing to system (2.16) in the form of a slow externally-applied sinusoidal current. This can be done by replacing the constant term I in the V-equation of (2.16) by a time-dependent function $I(\tau) = I_0 + \sin(\varepsilon\tau)$, with $\varepsilon > 0$ a small constant. However, to further analyse the resulting periodically forced system using geometric singular perturbation theoretical tools [79], it is more appropriate to write it in autonomous form and obtain the slow sinusoidal forcing $I(\tau)$ as the solution of a harmonic oscillator, that is, a second-order differential equation in I, which one can also write as a set of first-order differential equations in (I,J). Hence, we consider system (2.16) forced by the following slow differential equations:

$$\begin{aligned} I' &= -\varepsilon J, \\ J' &= \ \varepsilon(I - I_0). \end{aligned} \tag{2.17}$$

We are therefore considering the following extended 4D system:

$$\begin{aligned} CV' &= I - g_{\mathrm{L}}(V - E_{\mathrm{L}}) - g_{\mathrm{Na}} m_\infty(V)(V - E_{\mathrm{Na}}) - g_{\mathrm{K}} n (V - E_{\mathrm{K}}), \\ n' &= \frac{n_\infty(V) - n}{\tau_n(V)}, \\ I' &= -\varepsilon J, \\ J' &= \ \varepsilon(I - I_0), \end{aligned} \tag{2.18}$$

where the prime denotes differentiation with respect to τ which, in this context, is called the *fast time*. Hence, (2.18) is a slow-fast dynamical system with two fast variables V and n, and two slow variables I and J. As customary in multiple-timescale dynamics, we can rescale time by a factor ε and introduce the *slow time* $t = \varepsilon\tau$, which brings the system in a different time parametrisation that will be helpful when styding its slow singular ($\varepsilon = 0$) limit, namely:

$$\begin{aligned} \varepsilon C\dot{V} &= I - g_{\mathrm{L}}(V - E_{\mathrm{L}}) - g_{\mathrm{Na}} m_\infty(V)(V - E_{\mathrm{Na}}) - g_{\mathrm{K}} n (V - E_{\mathrm{K}}), \\ \varepsilon\dot{n} &= \frac{n_\infty(V) - n}{\tau_n(V)}, \\ \dot{I} &= -J, \\ \dot{J} &= \ I - I_0, \end{aligned} \tag{2.19}$$

where the over-dot denotes differentiation with respect to t.

As long as $\varepsilon \neq 0$, the systems (2.18) and (2.19) are equivalent, they have the same phase portraits, but the solution trajectories are parameterized differently. Furthermore, their respective singular limits are different and highlight different aspects of the original system's dynamics: the fast components in the case of system (2.18), and the slow components for (2.19).

The fast singular limit (i.e., system (2.18) with $\varepsilon = 0$) corresponds to the original integrator $I_{\mathrm{Na}}/I_{\mathrm{K}}$ model (2.16), which is logical since the extended system (2.18) was obtained by slowly forcing this integrator model. Recall that its bifurcation structure, shown on Fig. 2.2, is characterised by the presence of a SNIC bifurcation. The S-shaped curve of equilibria of this fast subsystem (2.16) is called *critical manifold* of the full system and we label it as S^0 in Fig. 2.2 and

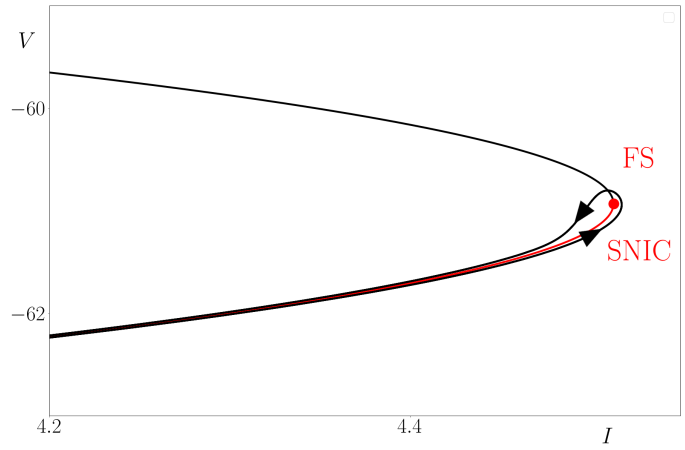

Figure 2.3: Folded-saddle canard trajectory (black curve with arrows) from (2.18) superimposed on the fast subsystem's bifurcation diagram already shown in Fig. 2.2. Arrow black curve: trajectory, red curve: unstable stationary points, black curve: stable stationary points. Parameter values are as in Fig. 2.2 except: $I_0 = 4$, $\varepsilon = 0.001$.

subsequent figures. Its algebraic expression is given by

$$S^0 := \left\{ I - g_L(V - E_L) - g_{Na} m_\infty(V)(V - E_{Na}) - g_K n_\infty(V)(V - E_K) = 0 \right\} \tag{2.20}$$

Hence, the critical manifold, which in the present case is a surface in $\mathbb{R}^4$ (since it has the dimension of the slow variables), can be seen as a graph over V and its equation can be written $S^0 = \{I = f(V)\}$. Figure 2.3 shows a trajectory of the slowly forced system (2.18) superimposed onto the critical manifold S^0 and displayed only in the vicinity of the fast subsystem SNIC bifurcation point, for specific initial conditions of the forcing $I(0) = I_0$ and $J(0)$. The trajectory (in black, with arrows representing the direction of motion) follows the stable branch of equilibria of the fast subsystem (red branch of the parabolic-shaped bifurcation curve), which is the behaviour predicted by slow-fast theory. However, near the SNIC point – also labelled FS for reasons related to the slow subsystem (see below) – the trajectory turns around the point, instead of being repelled away, before flowing backwards as the forcing changes direction. This behaviour is counter-intuitive, it has to do with certain types of canards, and it is best explained by considering the other singular limit, namely the slow limit, of the forced integrator system (2.18), as we do next.

Similar to the fast subsystem, we take the $\varepsilon = 0$ limit of system (2.19), which provides a good approximation of the slow dynamics of the forced integrator system. However, the slow singular limit is very different from the fast one and it is given by the following differential-algebraic system:

$$\begin{aligned} 0 &= I - g_L(V - E_L) - g_{Na} m_\infty(V)(V - E_{Na}) - g_K n(V - E_K), \\ 0 &= \frac{n_\infty(V) - n}{\tau_n(V)}, \\ \dot{I} &= -J, \\ \dot{J} &= \ I - I_0. \end{aligned} \tag{2.21}$$

The first two equations of (2.21) express the fact that in the slow singular limit, the dynamics is constrained to evolve only on S^0. In passing, this shows the importance of the critical manifold in both singular limits. The other two equations are the slow differential equations written in the slow-time parametrisation. The resulting system (2.21) is complicated to study as such, in particular because of the algebraic constraint which hides the limiting fast dynamics. However,

one can rescue it by differentiating this algebraic constraint with respect to time which, after projecting the dynamics onto the (V,J) plane (because the slow singular dynamics is essentially 2D) and rearranging terms, yields the following version of the slow subsystem:

$$\begin{aligned} f_V(V)\dot{V} &= -J, \\ \dot{J} &= I - I_0, \end{aligned} \tag{2.22}$$

with $I = f(V)$ due to the constraint to evolve on S^0 and where f_V denotes the derivative of f with respect to V, namely,

$$\begin{aligned} f_V(V) = g_{\mathrm{L}} + g_{\mathrm{Na}}(m_{\infty,V}(V)(V-E_{\mathrm{Na}}) + m_\infty(V)) + \dots \\ \dots g_{\mathrm{K}}(n_{\infty,V}(V)(V-E_{\mathrm{K}}) + n_\infty(V)), \end{aligned} \tag{2.23}$$

with:

$$x_{\infty,V}(V) = \frac{dx_\infty(V)}{dV} = \frac{\exp\{\left((V_{x,1/2}-V)/k_x\right)\}}{k_x} x_\infty(V)^2, \quad x = \{m,n\}.$$

System (2.22) is the more practical form of the slow subsystem or *reduced system (RS)*. This limiting system is singular along the zero set of f_V, which geometrically corresponds to the fold set $\mathscr{F} := \{f_V(V) = 0\}$ of the critical manifold. The critical manifold of system (2.19) is a cubic surface and its fold set $\mathscr{F}$ has two connected components; see, e.g., Fig. 2.5 for an illustration of the lower fold curve F of S^0. Noteworthy, the fold curve locally separates each of the two attracting sheets of S^0, along which $f_V(V) > 0$, from the repelling sheet defined by $f_V(V) < 0$. The attractiveness and repulsiveness of S^0 is inherited from the stability and instability of equilibria of the fast subsystem (2.16), given that its set of equilibria precisely corresponds to S^0. For a similar reason, the fold set $\mathscr{F}$ corresponds to the set of saddle-node bifurcations of the fast subsystem, hence the lower fold curve F corresponds to a family of SNIC bifurcation points of the fast subsystem. Namely, the SNIC point shown in Fig. 2.2 and detected when varying parameter I in (2.16) does not depend on J, hence we obtain a line of such point in the forced system (2.19).

In order to understand the flow of the RS near the fold curve F, one classical approach is to desingularise system (2.22) by rescaling time by a factor $f_V(V)$, which brings forth the so-called *desingularized reduced system (DRS)*

$$\begin{aligned} V' &= -J, \\ J' &= f_V(V)(f(V) - I_0), \end{aligned} \tag{2.24}$$

where the prime denotes differentiation with respect to the new time, i.e., after desingularization. System (2.24) is now defined everywhere on $\mathbb{R}^2$ including along the fold set $\mathscr{F}$.

Two points are worth noting about the DRS. First, the change of time by a factor $f_V(V)$ artificially creates in (2.24) the possibility for equilibria on the fold set $\mathscr{F}$, in particular on the lower fold curve F. Such equilibria satisfy the algebraic conditions

$$J = 0,\ f_V(V) = 0,$$

whereas the other equilibria of (2.24) satisfy the algebraic conditions

$$J = 0,\ f(V) = I_0,$$

and these are also equilibria of the RS system (2.22). Second, the same change of time, because of the V-dependent factor, makes the orientation of trajectories of the DRS be opposite to that of the RS whenever $f_V(V) < 0$, that is, along the repelling (middle) sheet of S^0. Hence, the DRS is a standard planar dynamical system, which can for instance have a saddle equilibrium on F, and the RS has the same geometrical orbits but the reversal of orientation along the repelling sheet

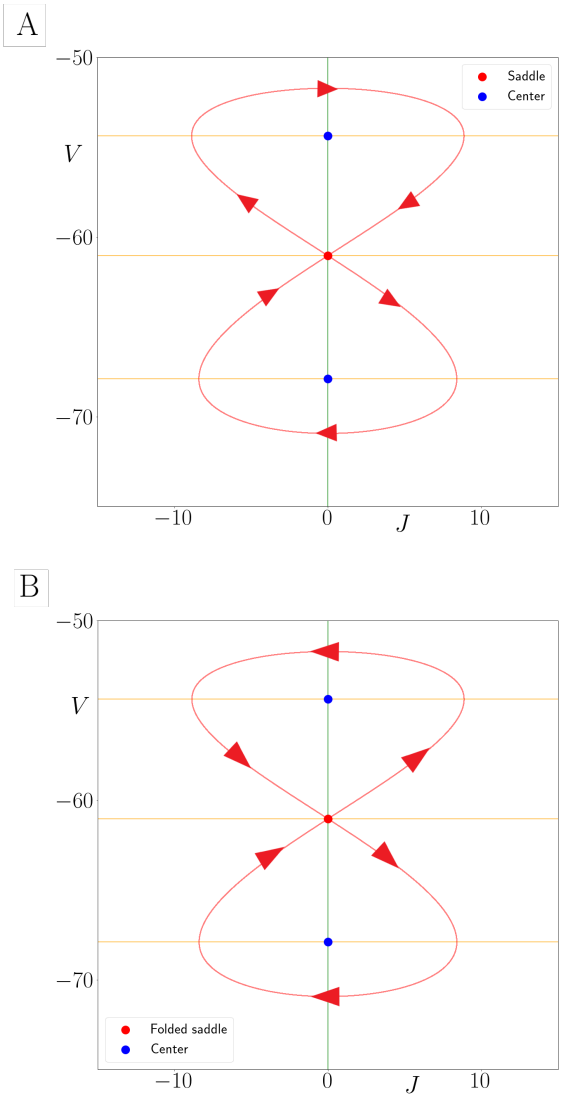

Figure 2.4: Phase portraits in the (J,V) plane. A: DRS system (2.24), B: RS system (2.16). In both panels, the orange lines are the components of the V-nullcline, while the green curve is the J-nullcline. A: the DRS has 3 equilibria, 2 centers (blue dots) and 1 saddle (red dot); the red curves are the stable and unstable manifolds of the saddle, they coincide and form a double homoclinic connection, each loop surrounding one of the centers. B: there are 2 center equilibria (blue dots) and one folded saddle (red dot). The red curves are the singular true and singular faux canards, they coincide to form a double folded homoclinic connection [27, 80]. Parameter values are as in the previous figures.

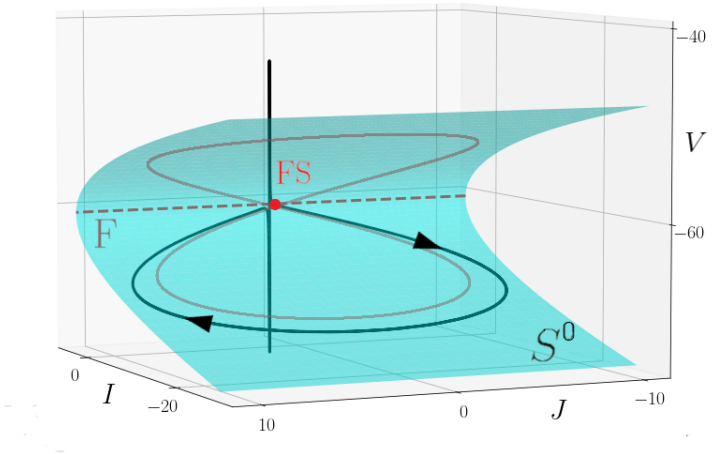

Figure 2.5: Phase portrait of system (2.18) projected onto the (I,J,V) space. Also shown is the critical manifold S^0 (blue surface), the lower fold curve F (dotted line), the singular true and faux canards (red curves), the folded-saddle singularity (red dot), labelled FS, and a trajectory making one spike without subthreshold oscillations (black curve). Parameter values are as in previous figures, initial conditions are: $V(0) = -76$, $n(0) = 3 \cdot 10^{-05}$, $I(0) = I_0 = -5.48$ and $J(0) = 10$.

of S^0 implies that the saddle equilibrium of the DRS is not an equilibrium anymore in the RS. Rather, it is a special point called *folded saddle*, which two special trajectories reach in finite time and cross. One trajectory crosses it from the attracting side of S^0 upwards, continuing along the repelling side, and it corresponds to the stable manifold of the saddle equilibrium of the DRS. The other trajectory crosses it in the opposite direction and it corresponds to the unstable manifold of the saddle of the DRS. Both trajectories are related to canards in that they cross from one side to the other of the critical manifold via a folded singularity. In the folded-saddle case, they are called *true singular canard* and *faux singular canard*, respectively [26].

Figure 2.4 shows the phase portrait of the DRS of system (2.19) on panel A, and of the corresponding RS's phase portrait on panel B. A saddle equilibrium of the DRS is located on F, therefore it corresponds to a folded saddle in the RS. The DRS has two other equilibria, both of center type, they are still (true) equilibria of the RS and, hence, they will also influence the dynamics of the full system. Figure 2.5 shows the RS's phase portrait in a 3D (I,J,V) projection, where the critical manifold S^0 is indeed a surface and F a curve; in fact, F is a straight line here since it does not depend on J. The figure illustrates well the geometry of such problems and the role of folded-saddle singularities in shaping the dynamics of slowly periodically-forced type-I neuron models. Indeed, such systems effectively correspond to *parabolic bursters* [81] and folded saddles organise the appearance of spikes in such bursters along solution branches in parameter space; see [27] for details.

On Fig. 2.5, a spiking solution of system (2.19) is shown on top of S^0 and it clearly appears that, as it reaches the (lower) fold curve F of S^0 and comes close to the folded-saddle point FS, the trajectory follows the true singular canard, then it makes a spike and, as the voltage is going down back to baseline, the trajectory follows the faux singular canard. Therefore in this context of type-I membrane model with slow periodic forcing, the spike-adding threshold is organised by folded-saddle canards [27]. Therefore, with a slow harmonic forcing, an integrator neuron like system (2.16) displays a folded-saddle singularity and still behaves as an integrator. In particular, it cannot generate subthreshold oscillations. This is essentially due to the eigenvalue ratio of the saddle equilibrium of the DRS, as we will see next.

2.4 Integrator neuron with resonator behaviour

2.4.1 The eigenvalue ratio of the DRS's saddle equilibrium

We have just seen that the slowly forced integrator neuron model has trajectories with no subthreshold oscillations. Our aim is now to create these subthreshold oscillations in order to effectively obtain a switch from integrator to resonator behaviour.

Recent results by Mitry and Wechselberger [53], also confirmed in the context of piecewise-linear slow-fast systems in [28], show that it is possible to obtain subthreshold oscillations near a folded saddle. More precisely, near the *faux canard* of a folded saddle, which is the perturbation for $\varepsilon > 0$ small enough of the singular faux canard described in the previous section and shown in Fig. 2.5. As proven in [53], the algebraic condition to obtain these subthreshold small-amplitude oscillations is on the ratio μ of the eigenvalues of the saddle equilibrium of the DRS, the ratio being of the unstable eigenvalue over the stable one. Necessarily μ is negative, however if it is strictly contained between -1 and 0, then such oscillations appear around the folded saddle's faux canard; see already Fig. 2.6 for an illustration.

As explained in the previous section, spiking trajectories of the forced system (2.19) follow the singular faux canard – hence, they also stay close to its ε-perturbation, the faux canard – right after making a spike, as the voltage goes down towards baseline. Hence, provided we can obtain subthreshold oscillations near the faux canard, then these will adequately resemble those obtained in resonator neurons, which will provide us with our objective of turning an integrator neuron into a resonator one purely based on a slow-fast effect.

Now, a rapid glance at the DRS (2.24) makes us conclude that its Jacobian matrix has zero trace, whatever the equilibrium solution around which one linearises. Hence, the saddle equilibrium of the DRS, which corresponds to the folded-saddle of system (2.19), is a neutral saddle, implying that the ratio of its eigenvalues is necessarily equal to -1. This will always be the case when slowly forcing an integrator model if the forcing is harmonic. Therefore, to obtain subthreshold oscillations one needs to consider a more elaborate forcing, namely one that includes a feedback term from the voltage.

2.4.2 Slow forcing with feedback from the voltage

We now consider a more general forcing, with a feedback term in V in the I equation in order to obtain a non-zero trace in the Jacobian matrix of the new DRS evaluated at the saddle equilibrium of interest. We will keep the J equation only dependent on I as we simply need one non-zero diagonal element in the Jacobian matrix evaluated at this saddle equilibrium in order to ensure that its eigenvalue ratio will be different than -1. For simplicity, we will keep the dependence in V in the I equation linear and show that it suffices to obtain the expected behaviour both at the level of the eigenvalue ratio and in the full system's solutions. Specifically, we define the new slow forcing (written in fast time) as

$$\begin{aligned} I' &= \varepsilon\,(-J+\alpha V)\,,\\ J' &= \varepsilon\ \ (I-I_0), \end{aligned} \tag{2.25}$$

which then yields the new DRS (after rescaling to the slow time)

$$\begin{aligned} V' &= -J+\alpha V,\\ J' &= f_V(V)(f(V)-I_0). \end{aligned} \tag{2.26}$$

Note that the voltage equation of the DRS is obtained from the slow differential equation for I in the forcing system, and that I must be kept equal to $f(V)$ in the slow singular limit. For these reasons, it would suffice that the I equation of the forcing depends on I and not on V, on top of its

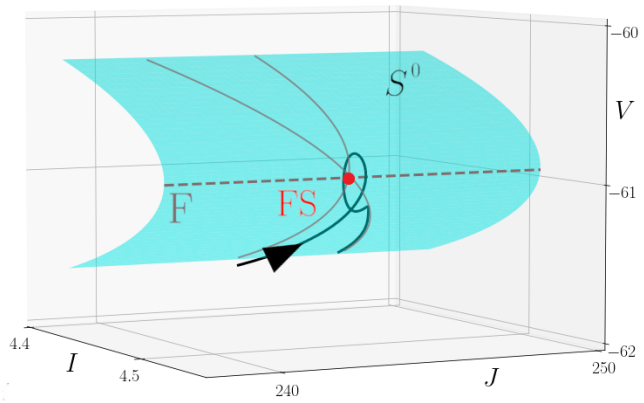

Figure 2.6: Phase portrait of system (2.19) projected onto the (I,J,V) space. Also shown at the critical manifold S^0 (blue surface), the lower fold curve F (dotted line), the singular true and faux canards (red curves), the folded-saddle singularity (red dot), labelled FS, and a trajectory making subthreshold oscillations (black curve). Parameter values are as in previous figures.

J dependence, in order to obtain a non-zero trace in the Jacobian matrix of the DRS corresponding to this new forcing. This would amount to replacing V by I in the first equation of (2.26). As a consequence, this in particular would avoid to use a feedback term in V in the full system (2.25). However, it turns out that the full dynamics would not be able to exploit within its spiking regime the eigenvalue ratio of the slow singular limit, as additional unwanted equilibria would arise.

Therefore we keep the new slow forcing system (2.25) to obtain the resonator behaviour. This new forcing may appear for now as the result of some *ad hoc* reverse engineering process, however we shall propose in the discussion section an explanation for its form and a possible experimental implementation of it.

We can now verify that the new DRS (2.26) does possess a saddle equilibrium on F and that one can take a value of parameter α so that its eigenvalue ratio is strictly between -1 and 0. At an equilibrium (V^*,J^*) located on the lower fold curve F of S^0, the Jacobian matrix of (2.26) reads

$$\mathbf{J} = \begin{pmatrix} \alpha & -1 \\ f_{VV}(V^*)(f(V^*)-I_0)) & 0 \end{pmatrix},$$

where $f_{VV}(V)$ is the second derivative of f with respect to V. Given that α contributes to the trace of $\mathbf{J}$ and not to its determinant, it is clear that we still have a saddle equilibrium on F, and hence a folded saddle in the full system with the new forcing. Therefore, the eigenvalue ratio μ is given by

$$\mu = \frac{\alpha + \sqrt{\alpha^2 - 4f_{VV}(V^*)(f(V^*)-I_0)}}{\alpha - \sqrt{\alpha^2 - 4f_{VV}(V^*)(f(V^*)-I_0)}}. \tag{2.27}$$

We then verify numerically that, for α negative and sufficiently large in absolute value, μ is indeed strictly between -1 and 0. For instance, by fixing $\alpha = -4$ one can observe the expect subthreshold oscillations when simulating the full system, as illustrated in Fig. 2.6 where we also show the critical manifold S^0 (blue surface), its lower fold curve F (dotted line), the folded-saddle singularity, labelled FS (red dot), and the two singular canards (red curves traced on S^0). Any values of α such that μ in expression (2.27) is between -1 and 0 will work equally well. The plotted trajectory is entirely subthreshold, however it comes close to the folded saddle and then turns back; as the

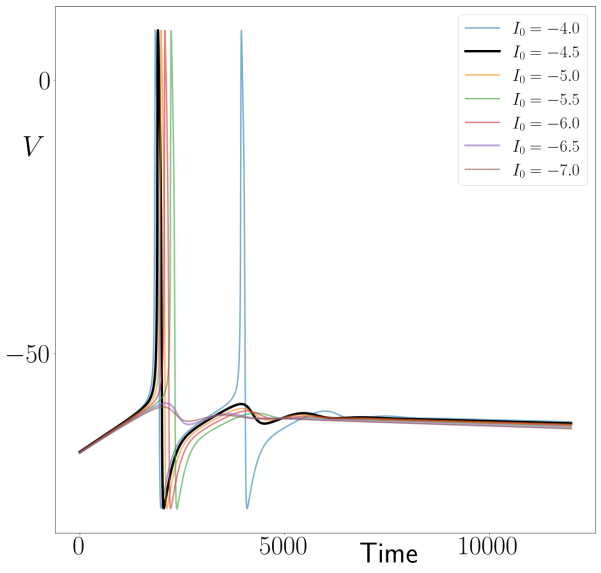

Figure 2.7: Integrator neuron (2.16) acting as a resonator with the slow forced current (2.25). Parameter values are as in previous figures except: $\varepsilon = 0.02$, $\alpha = -8$. Initial conditions are: $V(0) = -113.11$, $n(0) = 3 \cdot 10^{-05}$, $I(0) = I_0$ ranging between -4 and -7, and $J_0 = 546$.

voltage is going down towards baseline, it oscillates around the singular faux canard. Hence we have obtained a key feature of a resonator neuron by simply modifying the slow forcing received by an *a priori* integrator neuron without modifying its bifurcation structure obtained with constant forcing. Resonator models have also other features which we can as well recover here.

In Figure 2.7, we highlight this resonator effect in the time series of the full system with the new forcing, obtained by taking an ensemble of initial conditions for the forcing, namely varying I_0. What we observe is a transition in the voltage response from no spike, to one spike and then two spikes, depending on the value of I_0. Every trajectory with at least one spike has clear subthreshold oscillations after the spike, or after the second spike for trajectories that have two spikes. What is more, the values of I_0 at which we observe a first spike in the voltage response, and then a second spike, are quite specific. This confirms that the forced system resonates with specific inputs. Consequently, we have obtained the main features of a resonator neuron. Again, the main mechanism behind this switch of behaviour is purely due to the slow forcing received by the integrator model, and based upon multiple-timescale dynamical phenomena.

2.5 A multiple-timescale scenario for the reverse switch: from resonator to integrator

So far, we have mostly focused on the switch from an integrator to a resonator while retaining the characteristics of the integrator in absence of forcing. It is also possible to obtain the reverse switch, that is, to have a resonator neuron that behaves like an integrator. To do so, we keep system (2.16) but now consider yet another slow forcing, namely:

$$\begin{aligned} I' &= \varepsilon\left(-\beta J + \alpha(V - V_0)\right), \\ J' &= \varepsilon\ \ (I - I_0), \end{aligned} \tag{2.28}$$

where β is a new parameter that allows us to regulate the applied current; β was implicitly equal to 1 in the previous slow forcing (2.25). We also add V_0 as new parameter as it helps controlling the

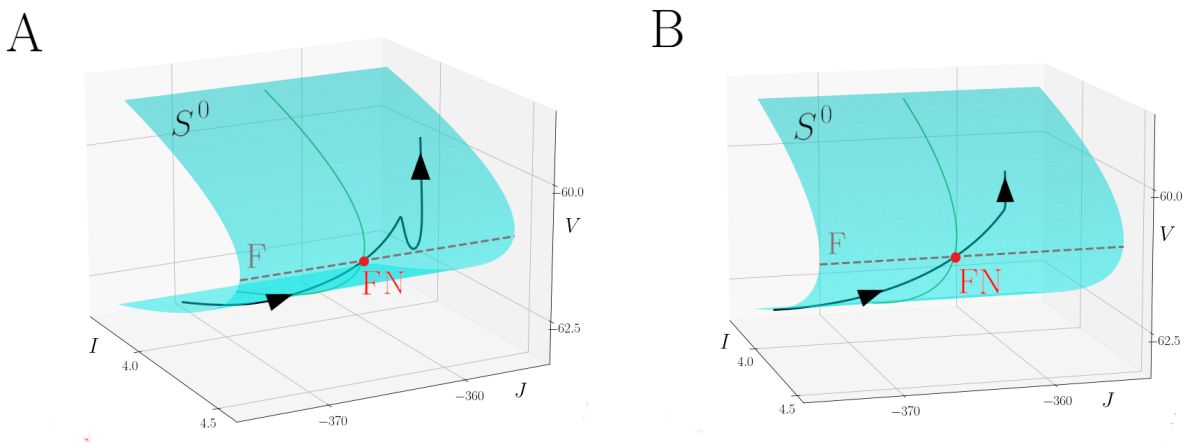

Figure 2.8: A: Resonator neuron model (2.16) with the slow forced current (2.28) for $J(0) = -370$. The folded node is labelled FN. B: Same model acting as an integrator for $J(0) = -380, I(0) = -6.0$. Parameter values are as before except: $\beta = -1, V_0 = -65.933$.

amplitude of the feedback term from the voltage; V_0 was implicitly equal to 0 in the previous slow forcing. The new parameter β also enables to change the type of folded singularity by changing the topological type of the DRS equilibrium located on the fold curve F. In particular, varying β may turn the saddle into a node, hence giving a folded node in the full system. The non-singular ($\varepsilon > 0$ small) dynamics near a folded node is well known to produce small oscillations, which in the neuronal context correspond to subthreshold oscillations [26]. Also known is the fact that, near a folded node, the phase space is locally partitioned into rotation sectors in which trajectories make a fixed number of subthreshold oscillations. Hence, it suffices to take initial conditions into the first rotation sector in order to obtain that the dynamics of this resonator system appears to behave like an integrator; Fig. 2.8 illustrates this effect.

Therefore, it is interesting to showcase this effect of a resonator that behaves like an integrator without changing anything to its structure. However, the observed behaviour has a notable discrepancy with standard resonator neurons, whereby the subthreshold oscillations occur before the spike and not after; see Fig. 2.8 A. Yet, this type of scenario can be related to experimental observations since there are neurons with this kind of electrical behaviour recorded *in vitro*; see [82] for an example in dorsal root ganglion neurons of rats.

2.6 Discussion

In this work, we have studied a novel mechanism for excitability switch, from integrator to resonator and *vice versa*, within the framework of multiple-timescale dynamical systems. As a proof-of-concept, we considered the simple yet biophysical 2D example of the I_{Na}/I_K model by Izhikevich, but it would work just as well in any type-I model. Starting from a parameter set in which the model behaves as an integrator in the classical sense, that is, where the spiking regime occurs through a SNIC bifurcation, we first applied a slow periodic forcing in order to create a multiple-timescale structure in the forced model, and to show that the resulting 4D model possesses a folded-saddle singularity. Following standard geometric singular perturbation theory (GSPT), we derived the desingularized reduced system (DRS) and showed that it had a saddle equilibrium on the fold curve of the critical manifold of the original system. This slow periodic forcing preserved the integrator behaviour. However, modifying the forcing by including a feedback term from the membrane potential, we managed to affect the eigenvalues of the saddle equilibrium of the DRS. This, according to recent results from GSPT [53], has the effect to allow for small-amplitude oscillations in trajectories that pass near the folded saddle. In turn, this provided us with a mechanism

to obtain subthreshold oscillations in a neuron model that was *a priori* an integrator, hence making it behave like a resonator. The feedback term in V in the slow current forcing can be seen as a simple form of *autaptic* connection, which is physiologically plausible.

Therefore, we can see these results as theoretical predictions that we would like to verify experimentally. From the standpoint of electrophysiological measurements on real neurons, the possibility to apply a current that depends in real time on the readout potential from the neuron can be obtained through a *dynamic-clamp* protocol [83, 84]. The experimental setup allows to inject currents to a cell that depends upon the measured voltage in real time. It can be used in conjunction with pharmacological blockade of e.g. one ionic channel of the cell and injecting back the corresponding current as a result of a computer simulation using the measured voltage. We are current working on validating our theoretical prediction using dynamic-clamp experiments.

From the modeling's point of view, adding this term αV in the slow differential equation of the forcing current was a way to modulate the determinant of the Jacobian matrix of the DRS, and therefore obtain subthreshold oscillations after a spike as well as modulate their number. Thus, it is possible to keep the bifurcation structure and the characteristics of an integrator neuron model while forcing it to display the specific behaviour of a resonator neuron.

This difference with the classical scenario of integrator neuron models can potentially have interesting fallouts in the study of information transmission between neurons. Indeed, by definition, an integrator neuron integrates the message and returns to the rest electrical potential without subthreshold oscillation. The fact that an integrator neuron can resonate like a resonator neuron shows that the transmission of information is more complex than a simple integration of a message given upstream. This is an interesting avenue for follow-up research on this topic.

We have also shown that the inverse scenario of switch from a resonator towards an integrator is also possible, by using a folded-node scenario in place of a folded-saddle one. However, the subthreshold oscillations obtained with a folded-node scenario occur before the spike rather than after, which is uncommon for resonator neurons. This may be associated with certain types of neurons (as reported in e.g. [82]) and to the dynamical phenomenon of *mixed-mode oscillations (MMO)* [26]. The most important aspect of this numerical experiment is that the resonance phenomenon does occur and it can be related to known properties of folded-node canards, in response to specific forcing inputs [85]. Yet, one can control the trajectories in such a scenario so that no subthreshold oscillation occurs as the solution flows past the folded node, which effectively makes the resonator behave like an integrator. We plan to verify experimentally this theoretical and computational prediction, first in neuromorphic analog circuits and then, using dynamic clamp, in real neurons.

Finally, in both the folded-saddle and the folded-node scenarios, the question of chaos can be pertinent as chaotic attractors have been shown to appear near a folded saddle [86] and near a folded node [26], respectively. Looking for such dynamics in the context of excitability switches due to either type of folded singularity goes beyond the scope of the present work but it is an interesting question for future work. In particular, the presence of chaos in brain activity has gathered substantial attention in the past few decades [87–90] and, in this context, the role of multiple timescales is still to be fully unravelled.

Chapter 3

Complex excitability and "flipping" of granule cells: an experimental and computational study

The results presented in this chapter have been submitted for publication: J. Danielewicz[1], G. Girier[1], A. Chizhov[2], M. Desroches[2], J.-M. Encinas[3] and S. Rodrigues[1,4], Complex excitability and "flipping" of granule cells: an experimental and computational study [5]. My contribution to this work was in the design and analysis of the model.

After having examined the mechanisms of neuronal excitability of integrator and resonator neurons from a purely theoretical point of view, we will now approach the study of a phenomenon obtained experimentally: thus, we will look here at an atypical behavior of granule cells immatures of the dentate gyrus. In order to study this phenomenon, we will develop a mathematical model to highlight this unusual behavior and, subsequently, the analysis of the model will be done using the same tools as those used in the previous chapter. We will also integrate the experimental data obtained by our colleagues in this chapter in order to give a complete context to the reader.

3.1 Introduction

Depolarization block - a silent state that occurs when a neuron receives excessive excitation - is a very important feature and is regarded to have pathological relevance for some brain disorders, including epilepsy and schizophrenia [91–93]. Furthermore, depolarization block in dopaminergic neurons was suggested to explain the therapeutic action of antipsychotic drugs [93]. Among all the mechanisms responsible for transition into depolarization block, the inactivation of voltage-gated sodium channels is believed to play a key role [94–96]. Subsequently, it has been shown that decreasing the sodium conductance pharmacologically causes dopamine neurons to go into DB with lower maximal frequencies at lower values of applied current, whereas augmenting this conductance with the dynamic clamp has the opposite effect [96, 97]. Dynamic clamp, a term for the various combinations of software and hardware that simulate these conductances, has proven to be a valuable tool for electrophysiologists for studying different excitability classes of cells [98].

The dentate gyrus (DG) is the input gate to the mammalian hippocampal formation and has been implicated in spatial navigation, response decorrelation, pattern separation and engram formation. Continual neurogenesis in the adult dentate gyrus produces new granule cells (GCs) that

[1] BCAM Basque Center for Applied Mathematics, Bilbao, Bizkaia, Spain
[2] MathNeuro Project-Team, Inria Centre at Université Côte-d'Azur, France
[3] Achucarro Basque Center for Neuroscience, Leioa, Bizkaia, Spain
[4] Ikerbasque, the Basque Foundation for Science, Spain
[5] HAL: https://hal.science/hal-04232000/

integrate into the hippocampal circuit by establishing synapses with existing neurons [12, 99–101]. Granule cells are the prominent neuronal subtype within the DG, and have been studied extensively from the perspective of their intrinsic response properties. GCs are characterized by their peculiar delayed and heterogeneous maturation. Most of them (85%) are generated postnatally. From the primary dentate matrix, neural precursors migrate to the dentate gyrus between embryonic day 10 and 14 where they differentiate into neurons [102, 103]. Neurogenesis reaches a peak at the end of the first postnatal week and is largely completed toward the end of the first postnatal month [104]. Interestingly, the dentate gyrus retains the capability to give rise to new neurons throughout life, although at a reduced rate [105, 106]. In adulthood, after being generated in the subgranular zone, immature GCs are incorporated into pre-existing circuits, thus contributing to improve several brain functions including learning and memory processes. During a transient period of maturation, new GCs exhibit intrinsic and synaptic properties distinct from mature GCs, potentially underlying the contribution of neurogenesis to memory encoding [8–14].

The vast repertoire of electrical activity displayed by neurons is the result of membrane-bound ion channels, each producing a distinct conductance that facilitates current flux through the membrane. These conductances may be static, or their magnitudes may be voltage- or ligand-dependent. The intrinsic firing properties and ionic conductances in GCs are thought to reflect their developmental stage and maturation level [8, 107, 108]. Among DG granule cells, input resistance (R_i), threshold current (I_{thr}), and firing patterns have been used as signatures of the degree of maturation and circuitry integration. During the first few days, postmitotic neurons remain in the proliferative subgranular zone and display very high input resistance (several gigaohms), because of the low density of K+ channels in the plasma membrane. Immature GCs also express voltage-dependent Na- and K-channels at a low density. Thus, depolarizing current steps elicit ”immature” action potential (single spikes with small amplitude and long duration) in current-clamp recordings [105, 106, 109]. Maturating GCs show a progressive decrease in input resistance and an increase in spike amplitude and frequency, suggesting that a deep rearrangement of voltage-operated and non-gated channels occurs at the same time.

In this study, we focused on exploring firing patterns and the transition to depolarization block of granule cells in the dentate gyrus by using dynamic clamp electrophysiological recordings. We applied dynamic clamp recordings to explore the diversity of neurons and to tackle their excitability by adding artificial sodium-like or potassium-like voltage-gated channels. This approach allowed us to describe for the first time a new electrophysiological phenomenon that we have called “flipping”.

3.2 Materials and methods

3.2.1 Animals and treatment

Four- to six-week old C57BL/6 mice were used for all procedures. Mice were housed at constant humidity and temperature with a 12-h light/dark cycle with food ad libitum. The use of animals for experimentation and the experimental procedures are included in the approved protocols M20_2022_129 (2022-2025); M20_2022_130 (2022-2025) that have been reviewed and approved by the ethics (CEID and CEIAB) committees of the UPV/EHU and the Diputación Foral de Bizkaia.

3.2.2 Preparation of brain slices

Mice were anestethized with isoflurane and decapitated. Their brains were quickly removed and placed in ice-cold artificial cerebrospinal fluid (ACSF) containing (in mM): 92 NMDG, 2.5 KCl, 0.5 $CaCl_2$, 10 $MgSO_4$, 1.25 NaH_2PO_4, 30 $NaHCO_3$, 20 HEPES, 5 Na-ascorbate, 3 Na-pyruvate, 2 Thiourea and 25 D-glucose (pH 7.3 – 7.4; 300 – 310 mOsm) and bubbled with the mixture of 95%

O2 – 5% CO2. Coronal slices (thickness = 250 µm) containing DG were cut from hemisphere ipsilateral to the TBI/Sham surgery using a vibrating microtome (Leica VT1000). Slices were stored submerged in room temperature for recovery.

3.2.3 Electrophysiology

After 1-1.5 h individual slices were placed in the recording chamber mounted on the stage of Scientifica microscope with 40x water immersion lens and superfused at 3 ml/min with warm (32 ± 0.5 °C), modified ACSF of the following composition (in mM): 124 NaCl, 4.5 KCl, 1.25 NaH_2PO_4, 26 $NaHCO_3$, 1 $MgSO_4$ * 7 H_2O, 1.8 $CaCl_2$, and 10 D-glucose (pH 7.3-7.4; 300-310 mOsm), bubbled with the mixture of 95% O2 – 5% CO2. Recording micropipettes were pulled from borosilicate glass capillaries (Science Products) using the PC-100 Nareshige puller. The pipette solution contained (in mM): 125 K-gluconate, 20 KCl, 2 $MgCl_2$, 10 HEPES, 4 Na_2-ATP, 0.4 Na-GTP, 5 EGTA (pH 7.3-7.4; 295-305 mOsm,). Pipettes had open tip resistances of approx. 7-9 MΩ. The calculated liquid junction potential using this solution was 13.1 mV, and data were corrected for this offset. Signals were recorded using Axon MultiClamp 700B amplifier (Molecular Devices), filtered at 2 kHz, and digitized at 20 kHz using Digidata 1550A (Molecular Devices) interface and Clampex 10 software (Molecular Devices, USA).

Cell access was obtained in the voltage-clamp mode and Resting Membrane Potential (RMP) was measured immediately upon break-in in the current-clamp mode by setting the clamp current equal to zero. In order to understand better the functional role of ion channels in shaping the electrical activity and entry to depolarization block granule cells were recorded under dynamic clamp conditions. In our specific setup, a second computer was connected to the MultiClamp 700B amplifier via the NI DAQ PCI-6221-37pin (National Instruments, Austin, TX). We employed a dynamic-clamp setup with open source software[6]. Contrary to alternative available dynamic clamp distributions, our software allows for a flexible development environment, which is the key requirement for later enhancements of specific experimental protocols. Once the two systems were connected, whole-cell recordings were performed in real-time, currents injected into the cell were directly dependent on measured voltage. The firing characteristics of the recorded cells was assessed using intracellular injections of rectangular current pulses of increasing amplitude (range:- 200 pA to + 1200 pA; duration: 500 ms) and f-I curves (firing rate vs injected current) were constructed. With dynamic clamp, we mimicked additional channels and assessed several parameters, like conductance, half-maximum location of the activation and inactivation functions of the ion channels responsible for general electrical activity and entry to DB of the recorded cells. To describe the additional currents, we employed conventional approximations such as Hodgkin-Huxley ones for potassium channels, and Markovian model for sodium channels. Only cells with stable access resistance were accepted for the data analysis. A schematic of our experimental set up is presented in Fig. 4.1.

In the experiments where additional shunting was mimicked, the injected current was calculated as a linear function of V:

$$I_{\text{inj}}(V,t) = u(t) - s(t)(V + 60\,\text{mV}), \tag{3.1}$$

where $u(t)$ and $s(t)$ are step functions of time. The reference membrane potential was choosen to be equal to -60mV.

In the experiments with the additional sodium and potassium channels, the injected current was as follows:

$$I_{\text{inj}}(V,t) = I_{\text{Na}}(V) + I_{\text{K}}(V) + I_{\text{KM}}(V) + u(t) - s(t)(V + 60\,\text{mV}), \tag{3.2}$$

[6] Available at http://www.ioffe.ru/CompPhysLab/AntonV3.html

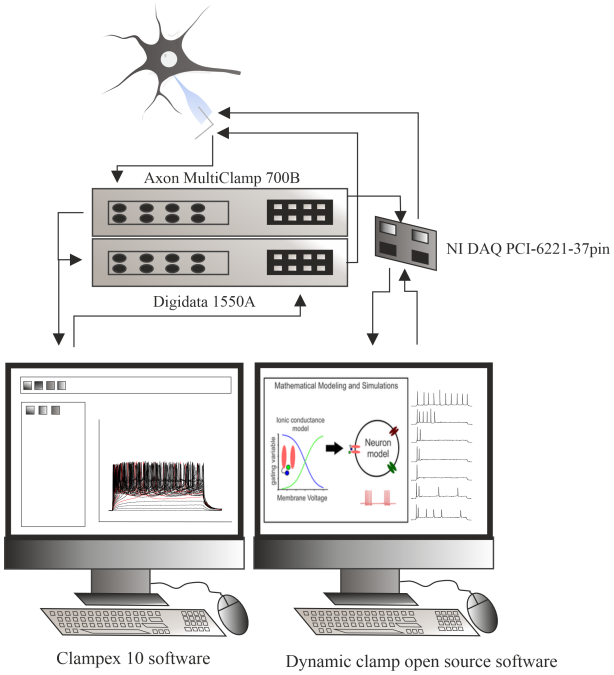

Figure 3.1: Dynamic-clamp experimental setup

where I_{Na}, I_{K} and I_{KM} were calculated according to ODEs coming from a mathematical model.

3.2.4 Statistical analysis

Statistical analysis was performed using the dynamic clamp open source software Visualizator[7] and GraphPad Prism[8]. Data were analyzed using a one–way ANOVA. Post hoc analysis for ANOVA was conducted by using Tukey's test. The level of statistical significance was set at $p < 0.05$. All data sets were tested for deviation from normal distribution (Kolmogorov–Smirnov's test).

3.2.5 Computational Modelling

We have combined multi-timescale mathematical modelling for excitability framework developed in [110] and conductance-based modelling.

The model's equations are defined as follows:

$$C\frac{dV}{dt} = g_L(V - V_L) - I_{Na} - I_{DR} - I_A + u(t) - s(t)(V - V_{us}) \tag{3.3}$$

Approximating formulas for the currents I_{Na}, I_{DR} and I_A are taken from [111]. The voltage-dependent potassium current I_{DR} is defined as:

$$I_{DR}(V) = \bar{g}_{DR} n y_K (V - V_K), \tag{3.4}$$

$$\frac{dn}{dt} = \frac{n_\infty(V) - n}{\tau_n(V)}, \tag{3.5}$$

$$\frac{dy_K}{dt} = \frac{y_{K\infty}(V) - y_K}{\tau_y(V)}, \tag{3.6}$$

[7] Available at http://www.ioffe.ru/CompPhysLab/AntonV3.html
[8] https://www.graphpad.com/

$$\begin{aligned}
\tau_n &= \frac{1}{\alpha_n+\beta_n}+0.8\,\text{ms}, \quad n_\infty = \frac{\alpha_n}{\alpha_n+\beta_n}\\
\alpha_n &= 0.17\cdot \mathrm{e}^{(V+5)\cdot 0.090}\,\text{ms}^{-1}, \quad \beta_n = 0.17\cdot \mathrm{e}^{(V+5)\cdot 0.022}\,\text{ms}^{-1},\\
\tau_y &= 300\,\text{ms}, \quad y_{K\infty} = \frac{1}{1+\mathrm{e}^{(V+68)\cdot 0.038}}
\end{aligned} \tag{3.7}$$

The voltage-dependent potassium current I_A is defined as:

$$I_A(V) = \bar{g}_A n_A^4 l_A^3 (V - V_K), \tag{3.8}$$

$$\frac{dn_A}{dt} = \frac{n_{A\infty}(V) - n_A}{\tau_n(V)}, \tag{3.9}$$

$$\frac{dl_A}{dt} = \frac{l_{A\infty}(V) - l_A}{\tau_l(V)}, \tag{3.10}$$

$$\begin{aligned}
\tau_{n_A} &= \frac{1}{\alpha_{n_A}+\beta_{n_A}}+1\,\text{ms}, \quad n_{A\infty} = \frac{\alpha_{n_A}}{\alpha_{n_A}+\beta_{n_A}}\\
\alpha_{n_A} &= 0.08\cdot \mathrm{e}^{(V+41)\cdot 0.089}\,\text{ms}^{-1}, \quad \beta_{n_A} = 0.08\cdot \mathrm{e}^{(V+41)\cdot 0.016}\,\text{ms}^{-1},\\
\tau_{l_A} &= \frac{1}{\alpha_{l_A}+\beta_{l_A}}+2\,\text{ms}, \quad l_{A\infty} = \frac{\alpha_{l_A}}{\alpha_{l_A}+\beta_{l_A}},\\
\alpha_{l_A} &= 0.04\cdot \mathrm{e}^{-(V+49)\cdot 0.11}\,\text{ms}^{-1}, \quad \beta_{l_A} = 0.04\,\text{ms}^{-1},
\end{aligned} \tag{3.11}$$

The voltage-dependent sodium current I_{Na} was approximated by the following 4-state Markov model [111]:

$$I_{Na}(V) = \bar{g}_{Na} x_1 (V - V_{Na}), \tag{3.12}$$

$$x_1 + x_2 + x_3 + x_4 = 1, \tag{3.13}$$

$$\frac{dx_i}{dt} = \sum_{j=0, j\neq 1}^{4} A_{j,i} x_j - x_i \sum_{j=0, j\neq 1}^{4} A_{i,j} \text{ with } i = 1,2,3, \tag{3.14}$$

$$\begin{aligned}
&A_{1,2} = 3\text{ ms }, A_{1,3} = f_1^{1,3}(V), A_{1,4} = f_1^{1,4}(V),\\
&A_{2,1} = 0, A_{2,3} = f_2^{2,3}(V), A_{2,4} = 0,\\
&A_{3,1} = f_1^{3,1}(V), A_{3,2} = 0, A_{3,4} = f_2^{3,4}(V),\\
&A_{4,1} = f_1^{4,1}(V), A_{4,2} = 0, A_{4,3} = 0,
\end{aligned} \tag{3.15}$$

$$\begin{aligned}
f_1^{i,j}(V) &= \left\{ \tau_{min}^{i,j} + \frac{1}{\mathrm{e}^{(V - V_{1/2}^{i,j})/k^{i,j}}} \right\}^{-1}\\
f_2^{i,j}(V) &= \{\tau_{min}^{i,j} + [(\tau_{max}^{i,j} - \tau_{min}^{i,j})^{-1} + \mathrm{e}^{(V - V_{1/2}^{i,j})/k^{i,j}}]^{-1}\}^{-1},
\end{aligned} \tag{3.16}$$

3.3 Results

3.3.1 Intrinsic properties, firing pattern and depolarization block

We studied the firing patterns and the capabilities of granule cells for entering DB under dynamic clamp conditions. Firing rate was assessed by constructing f-I curves for each individual neuron; see Fig. 3.2 B. Furthermore, the maximum frequency of generating action potentials was calculated as the number of spikes per stimulation time (500 ms). Because DB depends not only on the injected current but also on the extra (synaptic) conductance (additional shunting), we measured

the dependence of firing rate versus current and conductance (Fig. 3.2A), thus revealing the full domain of spiking in the plane of those input parameters, which can be re-interpreted in terms of excitation and inhibition in the following way:

$$I_{\text{inj}}(V,t) = v(t) - s(t)(V + 60\,\text{mV}) = G_{\text{E}}(t)(V_{\text{E}} - V) + G_{\text{I}}(t)(V_{\text{I}} - V), \tag{3.17}$$

that is,

$$u(t) = G_{\text{E}}(t)(V_E + 60\text{mV}) + G_{\text{I}}(t)(V_I + 60\,\text{mV}), \quad s(t) = G_{\text{E}}(t) + G_{\text{I}}(t), \tag{3.18}$$

where G_{E} and G_{I} are the excitatory and inhibitory synaptic conductances, respectively.

Entrance into DB was defined when the recorded cells started generating action potentials with half of the maximum spiking frequency, and the u_{DB}/G_{in} parameter was measured, where u_{DB} is the value of u that leads to DB, and G_{in} is the input conductance of neuron at the resting state, evaluated from responses to current step injection. Based on firing characteristics, specifically maximum frequency of generating action potentials, we have observed that recorded cells can be divided into two main groups:
1) LFC (low-frequency cells) generating action potentials with a maximum frequency lower than 12 Hz (9 cells out of 18; Fig. 3.3 A,C), and
2) HFC (high-frequency cells) generating action potentials with a frequency higher that 12 Hz (9 out of 18 cells; Fig. 3.3B,C).
The maximum spiking frequency reached by LFC cells was significantly lower when compared to HFC cells (7.5 ± 1.4 Hz vs 22.9 ± 2.1 Hz, $p < 0.0001$; see Fig. 3.3 D).

In LFC cells, the firing rate decreased for u_{DB}/G_{in} bigger then 56.4 ± 6.38 mV due to depolarization block and it was siginificantly lower than for HFC cells (79 ± 10 mV; $p < 0.05; F = 2.5; t = 1.9; df = 14$; see Fig. 3.3 E), which indicates that LFC neurons are entering into DB earlier than HFC cells. We have compared basic intrinsic properties of recorded LFC and HFC cells, such as action potential (AP) amplitude, resting membrane potential (RMP), the duration of action potential measured as AP half width, input resistance, threshold for generating AP and tau. There were no significant differences between two groups (see Table 3.1). Therefore, we suggest that the maximum firing rates of different neurons are mainly related to the current critical for entering DB.

	RMP (mV)	AP Amplitude (mV)	AP Half width (ms)	Treshold (mV)	R_i (MΩ)	Tau (ms)	n
LFC	-75,84±2,429	128,5±2,712	1,287±0,135	-37,78±3,401	284,1±27,54^{ns}	16,91±1,56	9
HFC	-76,26±0,847	131,9±1,525	1,101±0,038	-42,18±2,222	205,7±21,58	18,59±2,39	9

Table 3.1: Basic parameters of recorded neurons. Data presented as $mean \pm SEM$; there were no significant differences between the groups ($p > 0.05$ ANOVA). LFC - low frequency cells, HFC - high frequency cells, RMP - resting membrane potential, R_i - input resistance, n - number of recorded cells.

3.3.2 Effect of extra channels

We studied whether additional sodium and potassium channels would significantly affect both firing rates and entrance into the depolarization block of recorded granule cells. We implemented the addition of both sodium and potassium channels and assuming that the maximum firing rate is controlled by the currents that provide spike frequency adaptation, we decided to add slow potassium channels (KM-channels). As a result, LFC neurons started showing much higher values

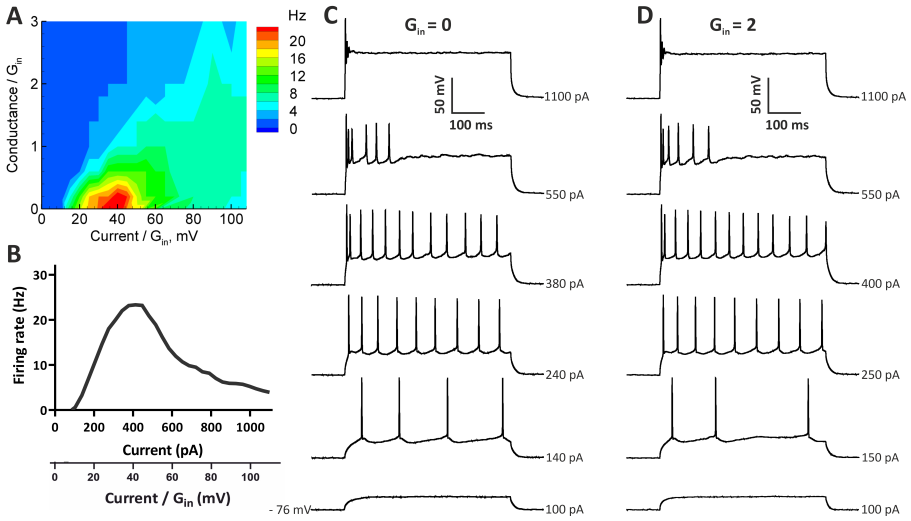

Figure 3.2: Assesment of firing rate in examplary neurons. (A) Firing rate as the number of APs per stimulation time interval (500ms) for a representative neuron as a function of injected current u and extra conductance s, both scaled by the input conductance G_{in}=9.9nS. Cell 21/C4. (B) Firing rate vs injected current steps (f-I curve) for cell 21/C4. (C) Representative responses of 21/C4 cell to sub- and suprathreshold depolarizing current pulses showing spiking pattern, entrance to DB and full depolarization block without extra conductane (G_{in}=0). (D) Representative responses of 21/C4 cell to sub- and suprathreshold depolarizing current pulses showing spiking pattern, entrance to DB and full depolarization block under shunting conditions (G_{in}=2)

of firing rate and the maximum frequency of generating action potentials was significantly higher when additional Na/KM channels were introduced (without extra channels 7.5 ± 1.4 Hz vs with Na/KM channels 36 ± 3 Hz; $p < 0.0001$; $F = 4.5$; $t = 8$; $df = 8$; Fig. 3.4 A,C). However, adding Na/KM channels did not significantly affected the u_{DB}/G_{in} parameter, therefore the entrance into DB of LFC cells has not been influenced (without extra channels 56 ± 6 mV vs with Na/KM channels 54 ± 9 mV; $p = 0.4$; $F = 1.6$; $t = 0.17$; $df = 11$; Fig. 3.4 D). In the HFC group, after adding extra Na/KM channels, a change to f-I curve previously observed in the LFC neurons was present only in 1 out of 9 recorded cells. In remaining HFC neurons additional Na/KM channels did not affected firing rate (Fig. 3.4 B,C). Overall, the mean maximum frequency of generating action potentials in HFC cells was not significantly altered by the addition of Na/KM channels (without extra channels 23 ± 2 Hz vs with Na/KM channels 36 ± 13 Hz; $p = 0.34$; $F = 30$; $t = 1.0$; $df = 7.5$; Fig. 3.4 B,C). In HFC group, after adding Na/KM channels, cells tend to enter DB block earlier than without additional channels as the value of the u_{DB}/G_{in} parameter is smaller, however this shift is not statistically significant (without extra channels 82 ± 11 mV vs with Na/KM channels 66 ± 18 mV; $p = 0.23$; $F = 2.3$; $t = 0.76$; $df = 10$; Fig. 3.4 D).

While analyzing f-I curves plots of recorded neurons we have observed an interesting phenomenon regarding depolarization block. In several granule cells, after adding Na/KM channels we observed a complex, non-gradual dependence of firing rate on injected current.

3.3.3 "Flipping" cells

While, in general, the initial entrance into DB was not affected by additional Na/KM channels as there were no significant differences in u_{DB}/G_{in} parameter (as described in the previous section), 8 out of 18 recorded neurons expressed a very unique behavior in terms of their firing patterns. After reaching a full DB characterized as generating only 1 or 2 action potentials in response to injected current step, these cells were not able to maintain it, instead they "flipped" and started generating trains of spikes at larger injected current steps before finally reaching another DB. We

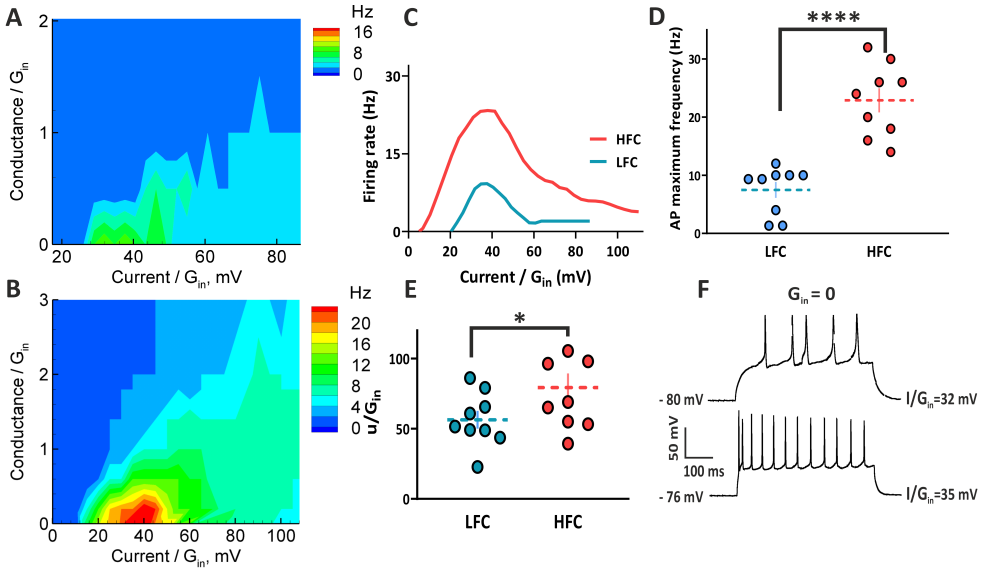

Figure 3.3: Comparison between two subpopulations of recorded cells. (A) Firing rate for a representative LFC neuron as a function of injected current and extra conductance, both scaled by the input conductance G_{in}=3.5nS. Cell 21/C1. (B) Firing rate for a representative HFC neuron as a function of injected current and extra conductance, both scaled by the input conductance G_{in}=9.9nS. Cell 21/C4. (C) f-I curves for representative LFC and HFC cells. (D) Maximum spiking frequency in LFC neurons was significantly lower than in HFC neurons. (E) LFC neurons are entering depolarization block earlier than HFC neurons. (F) Representative responses of LFC (top) and HFC (bottom) cells to suprathreshold depolarizing current pulse. Data presented as $mean \pm SEM$ (Horizontal dashed line); **** indicate statistical significance $p < 0.0001$ ANOVA; * indicate statistical significance $p < 0.05$ ANOVA

called this phenomenon "flipping" (Fig. 3.5). Interestingly, even within this small subpopulation of "flipping" cells we were able to observe two different types of "flipping" behavior. Half of the recorded cells were able to overcome the DB only once, meaning one big "flip" was present in their firing responses to current injection. However, the remaining half of the "flipping" neurons were able to produce multiple "flips" (2 to 8). In general, we observed that the majority (5 out of 8) of "flipping" neurons were previously assigned to the LFC group and within this subpopulation 2 cells were able to generate multiple "flips" (2 and 3 flips). The remaining 3 "flipping" cells were HFC neurons and 2 of those cells were able to generate 6 and 8 "flips" in response to injected current step (Fig. 3.6 A). As the majority of "flipping" cells generating only one "flip" were previously assigned to the LFC group, we wondered whether the number of "flips" is correlated with the initial spiking frequency. We have observed a positive correlation between the number of "flips" and the maximum frequency of action potentials generated by the cell in control conditions, meaning without additional Na/KM channels ($r = 0.74$; $R^2 = 0.54$; $p = 0.037$; Fig. 3.6 F), however there was no correlation between the number of generated "flips" and the maximum spiking frequency recorded with additional Na/KM channels present ($r = 0.24$; $R^2 = 0.055$; $p = 0.58$; Fig. 3.6 G). Because "flipping" is a phenomenon that we have never come across before, we wondered what could be the possible mechanism behind it, for this purpose we designed and analysed a computational model.

3.3.4 "Flipping" in a computational model

In order to analyze the "flipping" effect, we simulated a neuron with a Na- and K-channels. In the present model, the "flipping" is observed only for nonzero extra conductance. The excitation domain (Fig. 3.7 A) has a "horn" on the right. Hence, the f-I curve shows a gap (Fig. 3.7 C),

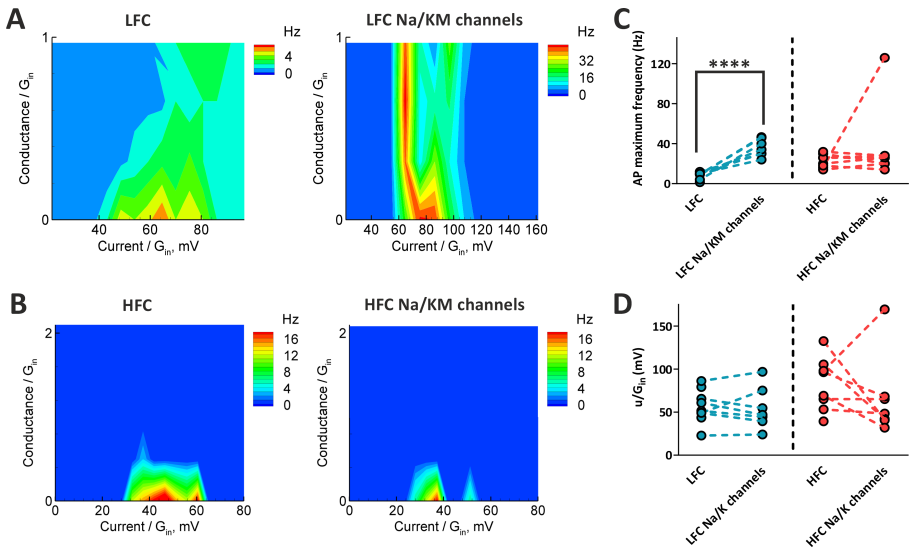

Figure 3.4: Effects of additional sodium and slow potassium channels on firing rate and enterance to depolarization block. (A) Firing rate for a representative LFC neuron without (left) and with additional Na/KM channels (right; g_{Na}=200 nS, g_{KM}=200 nS). (B) Firing rate for a representative HFC neuron without (left) and with additional Na/KM channels (right; g_{Na}=10 nS, g_{KM}=10 nS). (C) Maximum spiking frequency was significantly increased in LFC neurons, but not in HFC cells. (D) Addition of Na/KM channels does not influence the u/G_{in} parameter. Data presented as $mean \pm SEM$; **** indicate statistical significance $p < 0.0001$ ANOVA.

which is due to the "flipping". In terms of spike trains (Fig. 3.7 B), this gap is revealed as a cease of spiking for intermediate currents, and constant spiking in response to smaller and larger currents. At large currents, however, the amplitude of spikes is different, which is explained by a different sequence of transitions undergone by the sodium channels between the states of the Markov model. While during weak stimulus the hyperpolarization between spikes is stronger and the channels pass through low- and high-threshold states, for larger current the neuronal membrane is always depolarised between spikes and the channels pass only through high-threshold states. This recruitment of different channel states in different regimes causes the "flipping" behavior.

Blockade of potassium channels in the model leads to shrinkage of the excitation domain (Fig. 3.7 D), but remains the split and thus "flipping".

In the reduced model with only Na-channels present, the "flipping" is observed in wider range of extra conductances.

3.3.5 Bifurcation analysis of "flipping"

With the model introduced above, it is possible to reproduce the phenomenon of "flipping", and its appearance is linked to the number of states for the sodium channel of the model. With three states for this channel, the model displays one periodic regime upon variations of the applied current, and this regime is bounded in parameter space by Hopf bifurcations (one supercritical and one subcritical); see Fig 3.8. Namely, a family of low-voltage equilibria (rest states of the neuron) destabilise and give way to a family of stable limit cycles (spiking states of the neuron) via a supercritical Hopf bifurcation (HB_1), which occurs at an input current value $I \approx 1.2$. At a much higher value of the input current, the branch of stable cycles disappears via a subcritical Hopf bifurcation (HB_2) at $I \approx 2$. This scenario is compatible with type-2 neural excitability.

In contrast, when the number of states is increased to four, the two Hopf bifurcations from the previous scenario are still present, for similar values of applied current (HB_1, HB_2), however a second periodic regime appears, bounded in parameter space by a second pair of Hopf bifurcations,

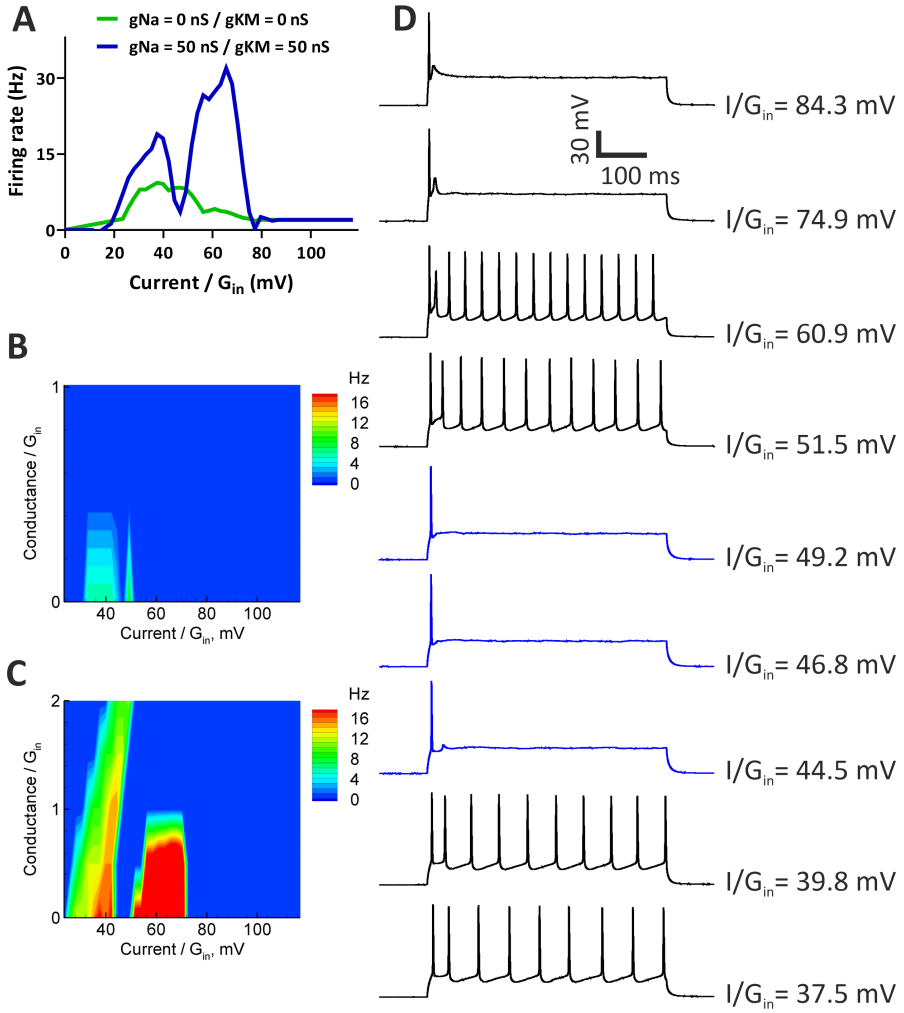

Figure 3.5: Example of the "flipping" phenomenon. (A) f-I curves of a representative cell exhibitng a pronounced "flip" in firing pattern induced by the addition of g_{Na}=50 nS, g_{KM}=50 nS channels (blue line). (B, C) Firing rate as a function of injected current and extra conductance, both scaled by the input conductance G_{in}=4.27nS without (B) and with (C) additional Na/KM channels. (D) Representative responses to a suprathreshold depolarizing current pulses.

at higher values of $I_{applied}$; see Fig 3.9. Indeed, at an input current value $I \approx 3.2$, the stable solution destabilise again and give way to a family of stable limit cycles via a supercritical Hopf (HB_3). This branch of stable cycles disappears via a subcritical Hopf bifurcation (HB_4) at $I \approx 4.1$. The two pairs of Hopf bifurcations are separated by a regime where the model admits a stable stationary state with voltage higher than the firing threshold, hence a DB. Therefore, this second periodic regime arising at larger values of the input current is therefore compatible with the "flipping" phenomenon reported in the present work, and it is due to the number of sodium channel states in the model.

3.4 Discussion

Managing the diversity of neurons is a very complex yet very important task. Electrophysiological criteria, such as intrinsic properties, excitability class represented by firing patterns and transition to depolarization block are very useful tools in identifying and distinguishing different types of

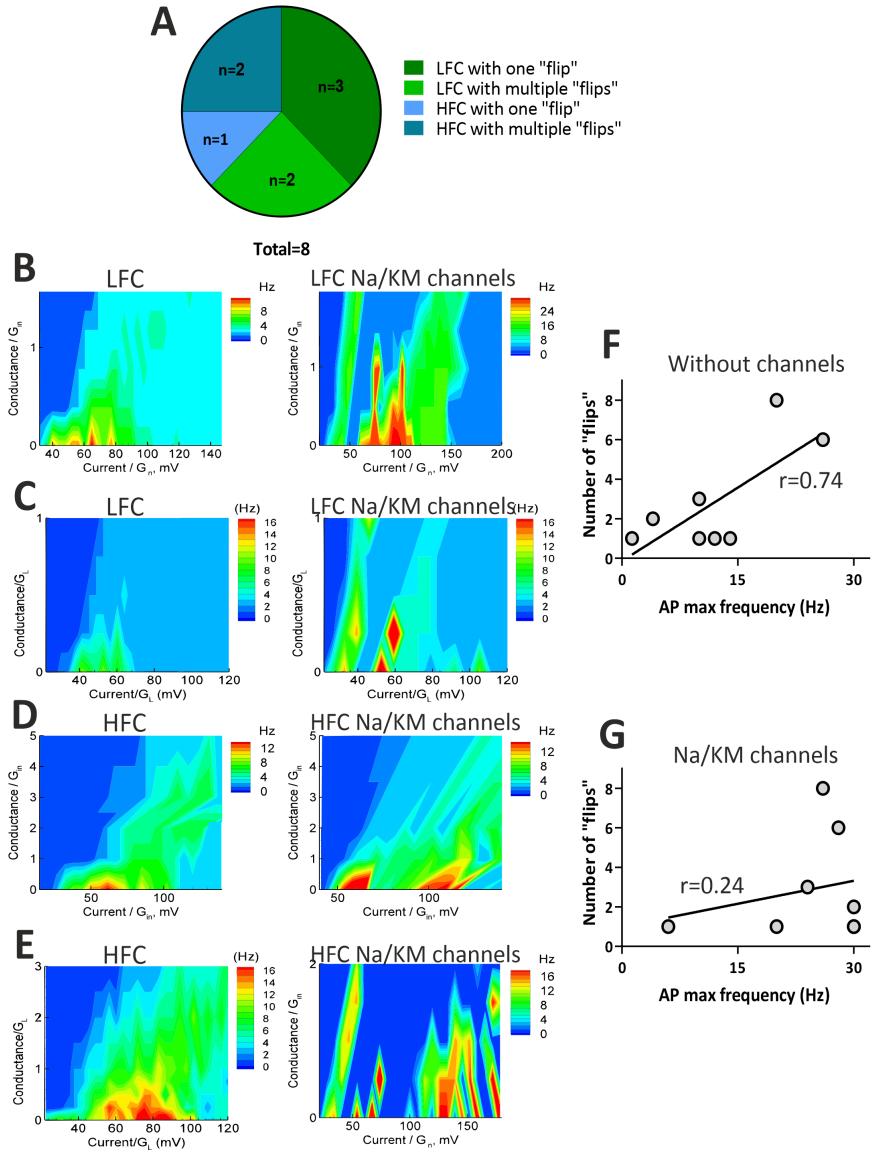

Figure 3.6: "flipping" cells. (A) Division of "flipping" cells based on the number of "flips" and the initial firing rate (LFC vs HFC). (B) Firing rate of a LFC neuron generating one "flip" after addition of g_{Na}=100 nS, g_{KM}=40 nS; Cell 21/C3. (C) Firing rate of a LFC neuron generating multiple "flips" after addition of g_{Na}=50 nS, g_{KM}=50 nS; Cell 21/C6. (D) Firing rate of a HFC neuron generating one "flip" after addition of g_{Na}=50 nS, g_{KM}=20 nS; Cell 21/C2. (E) Firing rate of a HFC neuron generating multiple "flips" after addition of g_{Na}=50 nS, g_{KM}=50 nS; Cell 21/C7. (F) The number of "flips" is correlated with the maximum frequency of spikes generated in response to injected current step before the addition of Na/KM channels. (G) The number of "flips" is not correlated with the maximum frequency of spikes generated in response to injected current step after the addition of Na/KM channels.

neurons that are sharing anatomical and functional similarities. Therefore, the primary goal of this study was to explore firing characteristics and entrance into depolarization block of granule cells.

Depolarization block of granule cells is very rarely discussed in experimental findings probably because the range of input currents in which GCs transition to DB could be considered unphysiological, especially in the case of mature GCs. However, a model developed for hippocampal CA1 region by Bianchi suggests that even background synaptic activity in the gamma range involving less than 3 percent of the total number of excitatory synaptic inputs converging on any

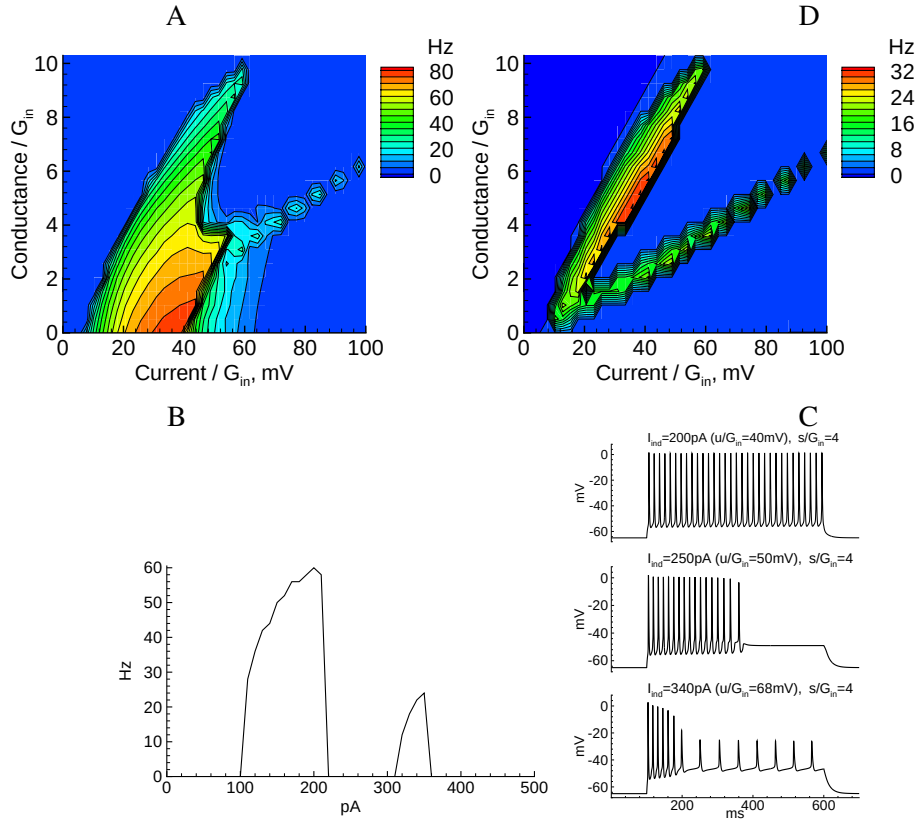

Figure 3.7: Simulations of a neuron with Na- and K-channels. g_{Na}=228nS, g_{KM}=500nS, G_{in}=5nS. A: f-u-s dependence; B: f-I curves for s/G_{in}=4; C: spike trains. D: Reduced model with only Na-channels.

given CA1 pyramidal neuron, could easily generate an aggregate peak input current larger than 1 nA [94]. This finding makes the DB a very important feature of any neuron, because it is highly probable that in large active networks, such as dentate gyrus, numerous cells will likely be in a DB at any time, therefore their silent state could influence the activity of the entire network.

Many studies focusing on the excitability of granule cells reported two main types of firing patterns that GCs generate in response to increasing current injections. Typically, type-1 DG granule cells are exhibiting spikes starting from a moderate intensity of stimulation and increasing in frequency following steps increments. At current injections of high intensity, these cells are lacking sustained firing, action potentials become progressively smaller in amplitude and GCs are entering DB at a relatively early stage (200-250 pA). In contrast, type-2 DG granule cells are usually described as firing throughout all the current step without failure (entering DB) and exhibiting firing frequencies that were linearly proportional to the intensity of the current step, i.e. the maximum number of action potentials was always observed for the highest current intensity [112–114]. Based on results obtained in our study, we observed that LFC neurons resemble the type-1 cells reported by others. In our experimental protocol, however, we used a wider range of current intensities therefore we were able to observe the transition of HFC (type-2) GCs into DB at a later stage (450-500 pA). That approach, with applying current intensities greater than 300 pA, is not typically used by others, therefore only linear firing patterns of type-2 GCs is most commonly described in other experimental findings.

Under physiological conditions, neurons fire in response to the activation of synaptic conductances. In electrophysiological experiments, usually, neuronal characteristics are probed in current-clamp conditions, which cannot fully reflect the synaptic activation because the injected current cannot mimick changes of membrane conductance, if only it is not voltage-dependent.

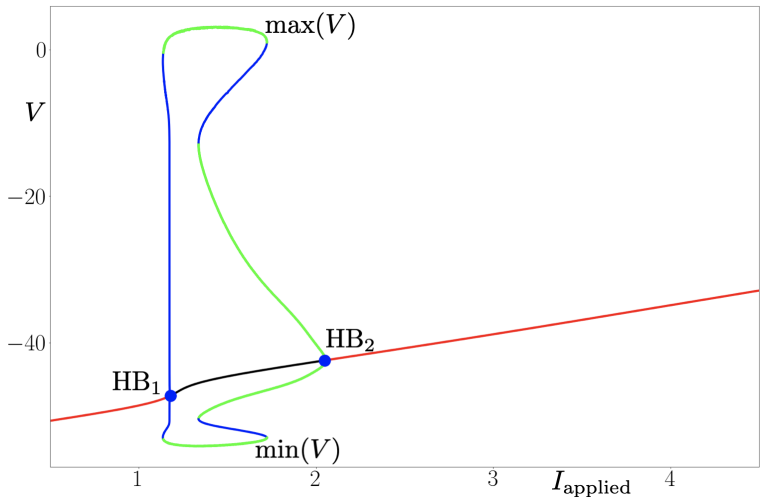

Figure 3.8: Bifurcation diagram of the system with 3 states, and with respect to parameter I_{applied}. Red (resp. black) segments of the curve of equilibria denote stable (resp. unstable) branches. As the applied current I_{applied} is increased, oscillations (spikes) appear through a subcritical Hopf (HB_1) and then disappear through a supercritical Hopf bifurcation (HB_2). Parameter values are: $\tau_{min}^{1,3} = 0.33$ ms, $\tau_{min}^{3,1} = 0.33$ ms, $\tau_{min}^{2,3} = 1.0$ ms, $V_{\text{half}}^{1,3} = -51$mV, $V_{\text{half}}^{3,1} = -42$mV,$V_{\text{half}}^{1,3} = -53$mV, $k^{1,3} = -2,0$ mV, $k^{3,1} = 1,0$ mV, $k^{2,3} = -1,0$ mV, $\tau_{max}^{2,3} =$ 100 ms, $V_{Na} = 65$ mV, $V_K = -70$ mV, $V_L = -64.96$ mV, $\bar{g}_{Na} = 2.28\mu S/cm^2$S, $\bar{g}_{DR} = 0.76\mu S/cm^2$, $\bar{g}_A = 8.36\mu S/cm^2$, $\bar{g}_L = 0.048\mu S/cm^2$, $s = 0.2cm^2$, $V_{us} = -49$ mV, $C = 0.7\mu$F/cm^2, $A^{1,2} = 3$ ms^{-1}. Initial conditions are: $V_0 = -65$, $x_1 = 0$, $x_2 = 0$, $n_0 = 0.00128$, $y_{K,0} = 0.47$, $n_{A,0} = 0.079$, $l_{A,0} = 0.85$.

The conductance change provides shunting effect, whose importance is shown, for instance, in visual cortex studies [115, 116]. Such characteristics of neuronal activity as the firing rate and the spike shape parameters are much more fully expressed by their dependence on both signals, the synaptic current and synaptic conductance, or, the injected voltage-independent current and conductance [117, 118]. These signals determine the first two, voltage-independent and linearly voltage-dependent components of the total current received by a neuron from synaptic input [117]. Therefore, in our study we used the dynamic-clamp technique, because it can provide both input signals, the synaptic current and the synaptic conductance [119, 120]. The dynamic-clamp stimulation protocols used in our study were based on [60, 120–123].

Every neuron has sodium and potassium currents and the interaction between these currents results in different transitions between spiking and the silent state, therefore here we studied the effects of additional Na- and KM-channels on firing rates and DB of GCs. In the case of rather "weak" LFC cells with a small maximum firing rates and narrow domain of spiking, the additional Na- and KM-channels increased the firing rate and enlarged the domain so that it resembled the firing pattern of HFC cells. Interestingly, in the case of HFC cells addition of Na/KM channels had no effect on maximum firing rate with the exception of one neuron. Moreover, the addition of extra Na/KM channels has no influence on the initial transition to depolarization block, except for "flipping" cells.

The discovery of the "flipping" phenomenon is the major finding of this study. As we have metioned earlier (see Results) we defined "flipping" behavior as the ability of certain neurons to overcome the initial depolarization block in order to start generating trains of spikes at larger injected current steps before finally reaching another depolarization block. To the best of our knowledge, this neuronal behavior has not been previously reported in other experimental studies.

Importantly, we were able to reproduce "flipping" phenomenon in our computational model reaching the conclusion that, the appearance of "flipping" is linked to the number of states for the

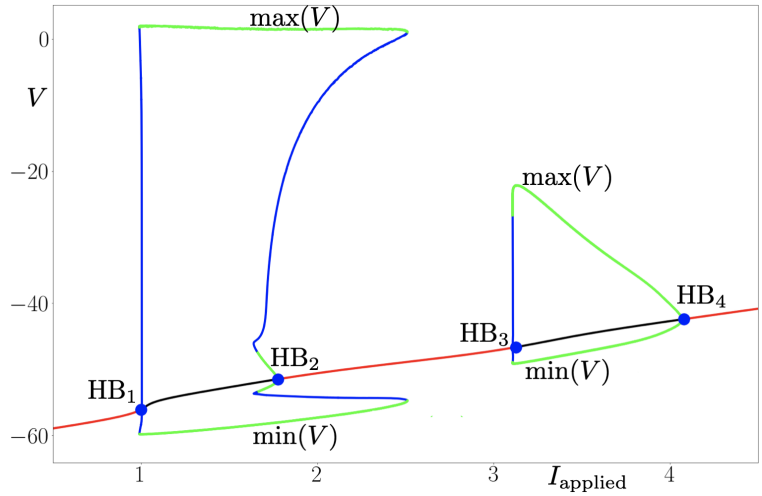

Figure 3.9: Bifurcation diagram of the system with 4 states, and with respect to parameter I_{applied}. Red (resp. black) segments of the curve of equilibria denote stable (resp. unstable) branches. As the applied current I_{applied} is increased, oscillations (spikes) appear through a subcritical Hopf (HB_1) and then disappear through a supercritical Hopf bifurcation (HB_2). Then, the scenario appears for higher values of I_{applied}: oscillations appear through a subcritical Hopf (HB_3) and then disappear through a supercritical Hopf (HB_4) Parameter values are: $\tau_{min}^{1,3} = 0.33$ ms, $\tau_{min}^{3,1} = 0.33$ ms, $\tau_{min}^{2,3} = 1$ ms, $\tau_{min}^{1,4} = 0.33$ ms, $\tau_{min}^{3,4} = 1ms$, $\tau_{min}^{4,1} = 0.33$ ms, $V_{\text{half}}^{1,3} = -51$mV, $V_{\text{half}}^{3,1} = -57$mV,$V_{\text{half}}^{1,3} = -53$mV, $V_{\text{half}}^{1,4} = -57$mV, $V_{\text{half}}^{3,4} = -60$mV, $V_{\text{half}}^{4,1} = -51$mV, $k^{1,3} = -2$ mV, $k^{3,1} = 1$ mV, $k^{2,3} = -1$ mV, $k^{1,4} = -2$ mV, $k^{3,4} = -1$ mV, $k^{4,1} = 1,0$ mV, $\tau_{max}^{2,3} = 100$ ms, $\tau_{max}^{3,4} = 100$ ms, $V_{Na} = 65$ mV, $V_K = -70$ mV, $V_L = -64.96$ mV, $\bar{g}_{Na} = 2.28\mu S/cm^2$, $\bar{g}_{DR} = 0.76\mu S/cm^2$, $\bar{g}_A = 8.36\mu S/cm^2$, $\bar{g}_L = 0.048\mu S/cm^2$, $s = 0.2cm^2$, $V_{us} = -49$ mV, $C = 0.7\mu$F/cm^2, $A^{1,2} = 3$ ms^{-1}. Initial conditions are: $V_0 = -65$, $x_1 = 0$, $x_2 = 0$, $n_0 = 0.00128$, $y_{K,0} = 0.47$, $n_{A,0} = 0.079$, $l_{A,0} = 0.85$.

sodium channel of the model. The voltage-dependent sodium current was approximated by the 4-state Markov model from [111], and its reduction to the 3-state version. The 4-state model model describes two closed, one open and one inactivated states of the channel. This version of the model shows a sharp threshold over a 5 to 10 mV range, depending on the recent history of the channel, and the ability to recover from inactivation during repolarizations positive to earlier thresholds, without a large window current, therefore the available threshold depends on the history of firing, shifting down during the course of spike repolarization without giving a strong window current.

With three states for the Na channel, the model displays one periodic regime upon variations of the applied current, and this regime is bounded in parameter space by Hopf bifurcations. This scenario describes a typical non-flipping cell. However, when the number of states is increased to four a second periodic regime appears, bounded in parameter space by a second pair of Hopf bifurcations, at higher values of applied current. The two pairs of Hopf bifurcations are separated by a regime where the model admits a stable stationary state resembling an initial depolarization block. The presence of the second periodic regime arising at larger values of the input current is compatible with the "flipping" phenomenon, therefore we believe that the apperance of "flipping" it is due to the number of sodium channel states.

Heterogeneities in intrinsic excitability and firing patterns of granule cells have been frequently reported in the context of neurogenesis. As we previously described (see Introduction) the intrinsic firing properties and ionic conductances in GCs are thought to reflect their developmental stage and maturation. There is a general agreement that GCs expressing less mature phenotype, previously described by others as type-1 cells, are reaching a maximum number of spikes with current steps of moderate intensity. At current injections of high intensity, these cells

are lacking sustained firing and are entering depolarization block. On the other hand GCs classified as type-2 are characterized by a linear firing in response to increasing current injections [113, 114]. It has been also recently reported that GCs located in different DG subregions exhibit different firing patterns [124]. Dentate gyrus, within each location along its dorso-ventral span, is anatomically segregated into three different sectors: the suprapyramidal blade, the crest region, and the infrapyramidal blade [125]. Across these sectors, granule cells manifest considerable heterogeneities in their intrinsic excitability, temporal summation, action potential characteristics, and frequency-dependent response properties. Having this in mind, the majority of our recordings were performed in the dorsal blade of the dentate gyrus within the crest. Therefore, it is not only possible, but more likely, that the two subpopulations of GCs described here (LFC vs HFC) are on a different stage of their maturation process, moreover we believe that the observed "flipping" phenomenon could be also correlated with neuronal maturation, especially when vast majority of "flipping" cells were LFC neurons. Immature neurons do not have a well-defined excitability identity (i.e. fixed conductivities) and rather have a fluid conductivity (akin to properties of stem cells) which enables them to explore the landscape of excitability types (1, 2 or 3). The final stage of maturation is stabilized via effectively being programmed by the environment set by local neuronal circuits in the DG.

We conjecture that our observed "flipping" behavior is induced by external electrical input (or in general electrical-chemical environment) that effectively pushes neurons to jump between classes of excitability. Without a proper "birth dating" technique (for example by using retroviral injections), that was not implemented here, unfortunately we cannot be sure. More research in the future will be required in order to describe "flipping" phenomenon in greater details and answer outstanding open questions: What is the possible role of "flipping"? Is "flipping" correlated with neuronal maturation? And finally, can this phenomenon be observed in other types of neurons in different brain regions?

Thus, the work presented in this chapter and the previous one shows the complementarity of models and experiments: in Chapter 2, we began our work with the mathematical study of a model, to then establish an experimental protocol applicable to a neuron *in vitro*. In this Chapter 3, our experimental colleagues produced a flipping phenomenon via an experimental protocol adapted to their setup, and we showed how this protocol also makes it possible to obtain the same result on a neuron model. This complementarity shows the robustness of the models as a representation of neurons, but also their importance as a tool.

Chapter 4

Observing hidden neuronal states in experiments

The results presented in this chapter has been submitted for publication as: D. Amakhin[1], A. Chizhov[2], G. Girier[3], M. Desroches[2], J. Sieber[4] and S. Rodrigues[3,5], Observing hidden neuronal states in experiments, arXiv: 2308.15477. My contribution to this work was the slow-fast analysis underpinning the reported phenomenon, as well as, the simulation of the protocols in-silico.

Let us now move on to a part of the thesis that combines mathematical, computational and experimental approaches. Chapters 4 and 5 focus on using the continuation method to study neuronal excitability in experiments, and so to bridge the gap between dynamical models, and real neuronal recording. Obtained from the continuation method, the *one-parameter bifurcation diagram* is a powerful representation to observe the excitability threshold, the spiking regions, but also the unstable states of a neuron. However, while the continuation method is well established for mathematical models for which the differential equations are known, its application to real experiments is more complex. In these chapters we will explore two alternative approaches to applying this method experimentally, one based on imposed current and the other on mastering the exact solutions of the experiments.

4.1 Introduction

When characterising the dynamics of nonlinear systems, one fundamental criterion for a model is if its invariants such as steady states or periodic orbits match experimental observations. The ability to validate models is, thus, greatly expanded by experimental tools with the capacity to unveil non-observable (sensitive or dynamically unstable) invariant states that are otherwise inaccessible to standard measurements. This thesis chapter applies the experimental technique to use feedback control for tracking unstable states while varying parameters to electrophysiology experiments on neuronal cells. Our aim is to support systematic validation of neuron models by comparing bifurcation diagrams and observing their between-cells variability. We focus on unstable parts of steady-state branches obtained by feedback-controlled experiments and compare them with indirect evidence from standard measurements from open-loop experiments. This extends recent work of Ori *et al.* [64, 126] constructing phase diagrams from neuronal data, and complements other approaches such as using data to verify the bifurcation structure of neuronal models [127,

[1]Laboratory of Molecular Mechanisms of Neural Interactions, Sechenov Institute of Evolutionary Physiology and Biochemistry of RAS, St. Petersburg, Russia
[2] MathNeuro Project-Team, Inria Centre at Université Côte-d'Azur; France
[3] BCAM Basque Center for Applied Mathematics, Bilbao, Bizkaia, Spain
[4]College of Engineering, Mathematics and Physical Sciences, University of Exeter, United-Kingdom
[5]Ikerbasque, the Basque Foundation for Science, Spain

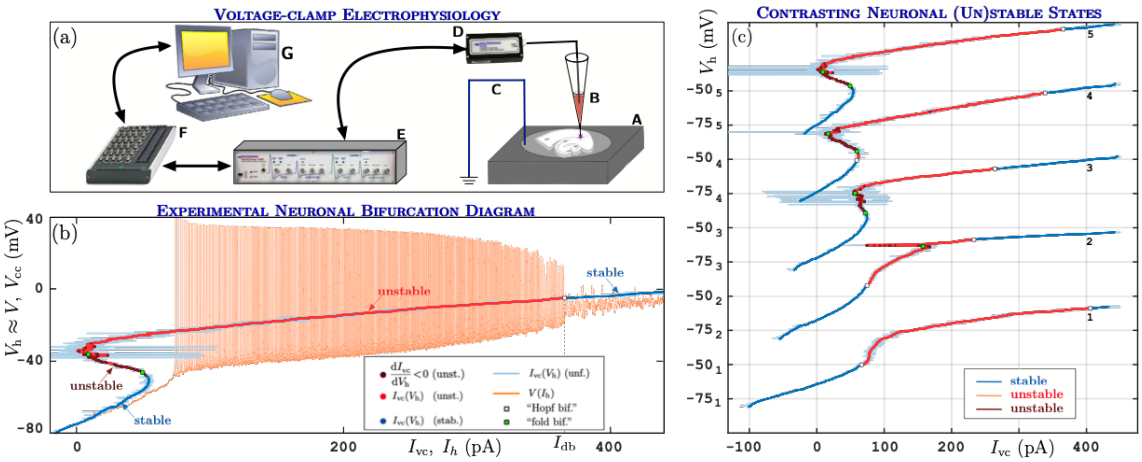

Figure 4.1: (a) Experimental setup with brain slice (A), patch pipette (B), reference electrode ((Ag-AgCl pellet) connected to ground (C), amplifier (E, *Multiclamp 700B*) with CV-7B headstage (D), AD-converter (F, *National Instruments NI USB-6343*) and standard PC computer (G). (b) VC and CC protocol runs for cell 5: $(I_{\rm vc}(t), \widetilde{V}(t))$-curve for VC run (thin bright blue: unfiltered data with sampling time step 5×10^{-5} s, blue/red/brown: median of $I_{\rm vc}$ over moving windows of size $\Delta w = 4 \times 10^3$ steps equalling 0.2s) and $(I_{\rm h}(t), V_{\rm cc}(t))$-curve for CC run (orange, mean of $I_{\rm h}$ over moving windows. (c) $(I_{\rm vc}(t), \widetilde{V}(t))$-curves for VC protocol of all 5 cells on waterfall $\widetilde{V}$-axis (same color coding as (b)).

128], model-based data analysis [129] or parameter estimation from data [130]. Our approach is similar to recent experimental demonstrations in mechanical systems [54], vibrations and buckling experiments [131, 132], pedestrian flow experiments [133], cylindrical pipe flow simulations [134] and feasibility studies for synthetic gene networks [135, 136].

In our electrophysiology experiments on entorhinal cortex neurons we apply a voltage clamp (VC) [3, 56], followed by current clamp (CC). In the VC setup the electrode acts as a voltage source, fixing the potential across the neural membrane, measuring the current, while the CC setup adds a fixed external current, measuring the resulting membrane potential (see Fig. 4.1(a)). The VC experiment is the closed-loop feedback-controlled part of the protocol and CC is the open-loop protocol, because in-vivo neurons are subject to current signals that drive spiking (oscillatory) or rest (steady) states of the neural membrane potential. VC has been applied successfully to study neuronal nonlinear current-voltage relationships, so-called *N-shaped I-V characteristics*, which cause enhanced neuronal excitability and influence the regenerative activation of certain ionic currents (e.g., sodium) [137–141] across the neural membrane, into and out of the cell. We show that the VC protocol with a slowly varied reference voltage signal gives access to stable and unstable neuronal steady states of the neuron. In contrast, the open loop CC protocol with a slowly varying applied current always follows stable (observable) states, driving the neuron to dynamically transition between its observable rest states and its observable spiking states.

We interpret these combined experimental protocols (VC and CC) using multiple-timescale dynamics, in particular, the dissection method [142], which reveals the dynamic bifurcations in the experiment; see Fig. 4.1(b). In this slow-fast framework, the states traced by the VC protocol with slow variation correspond to the steady-state experimental bifurcation diagram of the so-called *fast subsystem* of mathematical model describing the protocol [142]. Hence, the N-shaped I-V relation of a neuron should be seen as a S-shaped V-I bifurcation diagram; see again Fig. 4.1(b). Following this strategy, we demonstrate the feasibility of tracking a family of neuronal steady states (stable and unstable) via variations of reference signal and reparameterizing the obtained curve using the feedback current.

4.2 Material and methods

4.2.1 Animals and treatment

Male Wistar rats were used in this study (age P21, N=8 animals). For the bifurcation diagrams (see Fig.1b and Fig. 4.2), 4 of these 8 rats were used. Cells 1-3 were recorded from different animals, cells 4-5 were recorded from the same animal.

The use and handling of animals was performed in accordance with the European Community Council Directive 86/609/EEC. Horizontal 300-µm-thick brain slices were prepared as described in [143]. The slices contained the hippocampus and the adjacent cortical regions and were kept in the artificial cerebrospinal fluid (ACSF) with the following composition (in mM): 126 NaCl, 24 $NaHCO_3$, 2.5 KCl, 2 $CaCl_2$, 1.25 NaH_2PO_4, 1 $MgSO_4$, 10 glucose (bubbled with 95% O2/5% CO_2 gas mixture). All the listed chemicals were purchased from Sigma-Aldrich (St. Louis, MO, USA).

4.2.2 Electrophysiology

We performed the whole-cell patch-clamp recordings of the principal neurons in the entorhinal cortex. Neurons within the slices were visualized using a Zeiss Axioscop 2 microscope (Zeiss, Oberkochen, Germany), equipped with a digital camera (Grasshopper 3 GS3-U3-23S6M-C; FLIR Integrated Imaging Solutions Inc., Wilsonville, OR, USA) and differential interference contrast optics.

Patch pipettes were produced from borosilicate glass capillaries (Sutter Instrument, Novato, CA, USA) and filled with one of the following pipette solutions. A potassium gluconate-based pipette solution had the following composition (in mM): 136-K-Gluconate, 10 NaCl, 10 HEPES, 5 EGTA, 4 ATP-Mg, 0.3 GTP.

A Multiclamp 700B (Molecular Devices, Sunnyvale, CA, USA) patch-clamp amplifier, a NI USB-6343 A/D converter (National Instruments, Austin, TX, USA) and WinWCP 5 software (SIPBS, UK) were used to obtain the electrophysiological data. The recordings were lowpass filtered at 10 kHz and sampled at 20-30 kHz. The access resistance was less than 15 MΩ and remained stable during the recordings. The liquid junction potential was not compensated for. The flow rate of ACSF in the recording chamber was 5 ml/min. The recordings were performed at 30°C.

Specifically for the voltage-clamp protocol. We note the CV-7B headstage has four different feedback resistors (Rf): 50 MΩ, 500 MΩ, 5 GΩ, and 50 GΩ. The Rf determines the maximum currents that can be recorded or injected. In voltage-clamp mode it is generally recommended to use the largest possible value of Rf (larger Rf results in less noise), though high values can result in current saturation. Since in our preparation the electrical currents varied between 50-2000 pA (several nA for the potassium ion currents at positive holding potentials), we chose Rf= 500 MΩ (i.e. feedback gain $g_c = 2\,\mathrm{nS}$).

4.2.3 Equations and parameters for the simulations of the Morris-Lecar model

We applied the VC and the CC protocols, with slow variations in the feedback reference signal and in the applied current, respectively, to the Morris-Lecar model [6], whose equations are as follows

$$\begin{aligned} C\dot{V} &= -g_{\mathrm{L}}(V-V_{\mathrm{L}}) - g_{Ca}m_{\infty}(V)(V-V_{Ca}) - g_K w(V-V_K) + g_c(V_{\mathrm{h}}-V), \\ \dot{w} &= \phi\,\frac{w_{\infty}(V)-w}{\tau_w(V)}, \end{aligned} \tag{4.1}$$

with the following voltage-dependent (in)activation and time-constant functions:

$$\begin{aligned} m_\infty(V) &= 0.5(1+\tanh((V-V_1)/V_2)), \\ w_\infty(V) &= 0.5(1+\tanh((V-V_3)/V_4)), \\ \tau_w(V) &= \cosh((V-V_3)/(2V_4))^{-1}. \end{aligned} \tag{4.2}$$

To obtain Fig.1 of the main manuscript, we have used the following parameter values. Finally,

Table 4.1: Parameter values for the Morris-Lecar model (8.1).

parameter	C	g_L	V_L	g_{Ca}	V_{Ca}	g_K	V_K	g_c	ϕ	V_1	V_2	V_3	V_4
unit	pF	nS	mV	nS	mV	nS	mV	nS		mV	mV	mV	mV
value	20	2	-60	5	80	8	-74	20	0.067	-0.5	14	11	17.4

for both the VC protocol with slow variation, and the CC protocol with slow variation, the speed of the variation was chosen to be equal to $\varepsilon = 0.01$. Note that g_c can be decreased to 7, which is in the same order of magnitude as the experimental one. Moreover, the capacitance chosen for the model simulations is on the same order of magnitude as observed in the experiments. Nevertheless, the key point to note is that we are not aiming for quantitative agreement since this is a conductance-based phenomenological model.

4.3 Results

We applied first VC and then CC protocol to 5 neurons in the entorhinal cortex of 4 male Wistar rats [143]. We first performed the VC neuronal recordings varying the hold voltage $\widetilde{V}$ from $-80\,\mathrm{mV}$ to $+30\,\mathrm{mV}$ slowly with $\dot{\widetilde{V}} = 1.83\,\mathrm{mV/s}$, while measuring current, called I_{vc} in Fig. 4.1(b,c). Subsequently, for the CC recordings we first determine the minimal injected current required to induce a depolarization block of action potential generation (I_{db} in Fig. 4.1(b), upper limit of current input where firing occurs). Then we gradually increased the injected current from $0\,\mathrm{pA}$ to I_{db} during $60\,\mathrm{s}$, such that $\dot{I}_{\mathrm{h}}$ is in the range $6.6\ldots7.4\,\mathrm{pA/s}$ for the 5 neurons, while recording voltage $V_{\mathrm{cc}}(t)$.

Figure 4.1(b) shows the time profiles $(I_{\mathrm{h}}(t), V_{\mathrm{cc}}(t))$ of the CC protocol run (orange, thin) and $(I_{\mathrm{vc}}(t), \widetilde{V}(t))$ of the VC protocol run (bright blue, thin) for cell 5 overlaid in the (I,V)-plane. After smoothing the VC time profile is the S-shaped curve $(I_{\mathrm{vc,sm}}(\widetilde{V}), \widetilde{V})$ (blue/brown/red, thick). It equals the (I,V)-characteristic of the stationary neuronal states of the CC protocol, *including dynamically unstable states* (brown and red). The transition to stable spiking states is compatible with type-I excitability, however we do not have sufficient data to conclude. Their dynamical instability is inferred in two ways: (i) by the negative slope of the $(I_{\mathrm{vc,sm}}, \widetilde{V})$-curve (brown) after smoothing over a moving window with larger size $\Delta w = 1.5 \times 10^4$ steps ($= 0.75$s), or (ii) by the presence of oscillations (neuronal firing) in the CC run at I_{h} equalling the $I_{\mathrm{vc,sm}}$ (red). The stability boundaries of the stationary states are labelled as bifurcations in Fig. 4.1(b). The change of stability near the disappearance of stable spiking states at I_{db} is a Hopf bifurcation. The fold points of the $(I_{\mathrm{vc,sm}}, \widetilde{V})$-curve are saddle-node (fold) bifurcations. Figure 4.1(c) shows the stationary-state curves with their inferred dynamical stability for all 5 cells. The cells are vertically ordered and numbered according to depth of the S-shape, determining if they fall into the class of type-I or type-II neurons. Figure 4.1(c) demonstrates wide variability in steady-state curve shape among cells of nominally same function.

Figure 4.2 presents the recordings for VC and CC protocol for cells 1–4 in the same way as Fig. 1(b) (which was for cell 5). The graphs superimpose the observations of our VC protocol

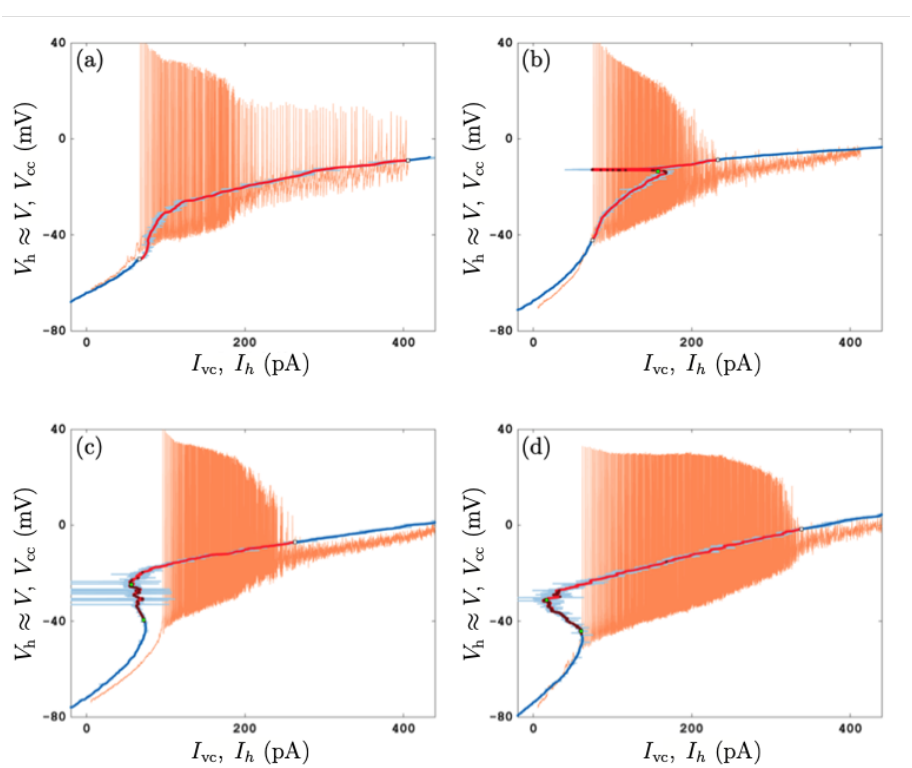

Figure 4.2: Experimental bifurcation diagrams for cells 1–4.

with slow variation of the feedback reference signal and the CC protocol with slow variation of the externally applied currents. The graphs confirm (as does Fig. 1(c) from the main manuscript) that type-II properties of the neurons vary between individual cells. We also observe that in all cells for $I_h > I_{db}$ and for cell 3 for small I_h the distance between the stable steady states obtained from the VC protocol and the corresponding trajectory segments obtained from the CC protocol is larger than the bias caused by dynamic variation of $\widetilde{V}$ and I_h respectively, based on estimate (5) of the main text. We do not have a complete physiological explanation for this mismatch, but we hypothesise that this is due to the natural physiological drift of underlying neuronal processes as one performs successive recordings on the same cell.

A characteristic for the class of excitability is the dependence of the spiking frequency on the external current I_{ext} as I_{ext} approaches the lower limit of the spiking region, with a drop in frequency expected for type-I neurons. Fig. 4.3 presents the interspike interval as a function of the hold current I_h in panel (a), and the frequency against I_h in panel (b) for the 5 cells for which experimental bifurcation diagrams are shown in Figure 1 of the main manuscript and Fig. 4.2. We detected crossing times for $V_{cc}(t)$ through the threshold $V = 0$ from below (or equivalently Poincaré map, at $V = 0$). As visible in Fig. 4.3, even though the 5 cells have different steady-state curves (obtained with the VC protocol), their frequency behaviours during the CC protocol are qualitatively similar with drops from ≈ 15 Hz to $\lesssim 5$ Hz for all cells. Hence, the frequency drop does not allow us to conclusively distinguish between different excitability classes for the present cells. However, the steady-state bifurcation diagrams obtained through VC protocols show that cells 3-5 exhibit an experimental bifurcation diagram that is compatible with type-1 excitability, while cells 1 and 2 have type-II compatible steady-state curves.

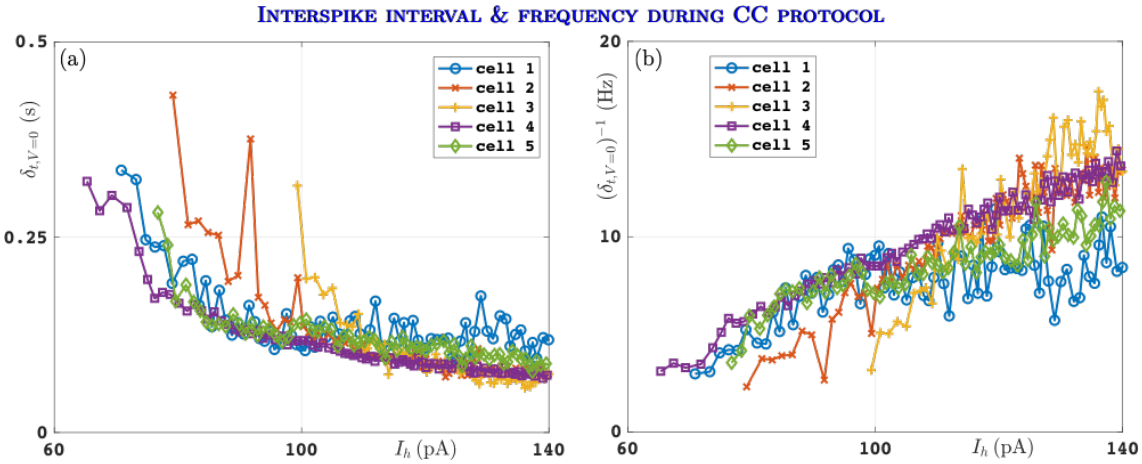

Figure 4.3: Interspike intervals in seconds (panel (a)) and frequency in Hz (panel (b)) for the spiking responses during the CC protocols with slowly-varying applied current near the low-current limit of the spiking region.

4.4 Analysis

To see why VC-run time profiles approximate the experimental bifurcation diagram with unstable stationary states of neurons in CC protocol, we use the formalism of multiple-timescale dynamical systems. Superimposing the data from VC and CC protocol also gives the first experimental illustration of slow-fast dissection. The effect of the respective clamps can be understood in a general conductance-based model for the neuron,

$$\begin{aligned} C\dot{V} &= -\textstyle\sum_j I_j(x_j,V) + I_{\text{ext}}, \\ \dot{x}_j &= \frac{x_{\infty,j}(V) - x_j}{\tau_j(V)}, \end{aligned} \tag{4.3}$$

describing the current balance across the neuron's membrane. The membrane potential is V, $I_j(x_j,V)_{j=1,\ldots,N}$ are the currents across different voltage-gated ion channel types and I_{ext} is the external current. Each ionic channel type j has an associated gating variable x_j with steady-state gating function $x_{\infty,j}(V)$ and relaxation time $\tau_j(V)$. The observed dynamic effects such as oscillations (firing/spiking) and negative-slope (I,V) characteristics are determined by these channel coefficients I_j, $x_{\infty,j}$ and τ_j that are traditionally obtained by parameter fitting from VC experiments, a difficult and mostly ill-posed problem [144].

The VC and CC protocols use different mechanisms for generating $I_{\text{ext}}(t)$. The VC protocol is a closed loop where a voltage source regulates I_{ext} with high gain g_c to achieve the slowly varying hold voltage $\widetilde{V}$ at the voltage source for (4.3), measuring I_{vc}:

$$\begin{aligned} I_{\text{ext}} &\approx I_{\text{vc}} = g_c(V - \widetilde{V}), \\ \dot{\widetilde{V}} &= \varepsilon\Delta_V, \end{aligned} \tag{4.4}$$

which turns (4.3), (4.4) into a multiple-timescale dynamical system with $N+1$ fast state variables (V,x_j) and one slow state variable $\widetilde{V}$, corresponding to the feedback reference signal [34]. The speed at which $\widetilde{V}$ varies is $\varepsilon\Delta_V$ with $\Delta_V(t) = 0.183\,\text{mV/ms}$, where we extract the dimensionless small factor $\varepsilon = 10^{-2}$.

In contrast, the CC protocol holds I_{ext}, measuring the generated voltage V_{cc}, thus, corresponding to an open-loop system, permitting e.g. the spiking seen in Fig. 4.1(b):

$$\begin{aligned} I_{\text{ext}} &\approx I_{\text{h}}, \\ V_{\text{cc}} &\approx V, \\ \dot{I}_{\text{h}} &= \varepsilon\Delta_I. \end{aligned} \tag{4.5}$$

The applied hold current I_{h} is varied slowly at speed $\varepsilon\Delta_I$ with $\varepsilon = 10^{-2}$ and $\Delta_I = 0.66\ldots0.75\,\text{pA/ms}$. System (4.3), (4.5) is also a slow-fast system with $1+N$ fast variables (V, x_j) and 1 slow variable I_{h}. The gain g_{c} in (4.4) is limited by the imperfect conductance across the non-zero spatial extent of the membrane. Even though (4.3) is for the potential V across the entire membrane and only $\widetilde{V}$ at the clamp is measured, we approximate $\widetilde{V}, V_{\text{cc}} \approx V$ for, the membrane potential, and $I_{\text{h}}, I_{\text{vc}} \approx I_{\text{ext}}$ for the external current in (4.3).

Following a classical multiple-timescale approach, we consider the $\varepsilon = 0$ limit of system (4.3), (4.4), which corresponds to its $(1+N)$-dimensional fast subsystem (4.3) with $I_{\text{ext}} = g_{\text{c}}(V - \widetilde{V})$, where $\widetilde{V}$ is now treated as a parameter. For fixed $\widetilde{V}$ and voltages in the range $-80\ldots30\,\text{mV}$ of interest, system (4.3) has only stable steady states (no limit cycles, that is, no neuronal spikes). For a fixed hold voltage $\widetilde{V}$, the steady states of model (4.3) with $I_{\text{ext}} = g_{\text{c}}(V - \widetilde{V})$ satisfy the algebraic equations for $(V, x_j)_{j=1,\ldots,N}$:

$$\begin{aligned} \textstyle\sum_j I_j(x_j, V) &= g_{\text{c}}(V - \widetilde{V}), \\ x_j &= x_{\infty,j}(V), \end{aligned} \tag{4.6}$$

where $I_{\text{eq}}(V) = \sum_j I_j(x_{\infty,j}, V)$ is the equilibrium current for fixed membrane potential V. The solutions of (4.6) form a (1D) steady-state curve of system (4.3), (4.4), which is normally hyperbolic (transversally attracting) for $\varepsilon = 0$. For $\varepsilon \neq 0$ the increase of $\widetilde{V}$ with speed $\varepsilon\Delta_V$ introduces a slow variation of all states (V, x_j). Hence, V and the feedback current $I_{\text{vc}} = g_{\text{c}}(V - \widetilde{V})$ (as measured), are not at their steady-state values given by (4.6), but they are changing dynamically. This results in a difference between the measured curve $(I_{\text{vc}}, \widetilde{V})$ in Figure 4.1(b) and the (I, V)-values of the desired steady-state curve (4.6).

Geometrical singular perturbation theory (GSPT) by Fenichel [79] implies that after decay of initial transients every trajectory of the VC protocol modelled by system (4.3), (4.4) satisfies equation (4.6) for the steady-state curve up to order ε. The first-order terms in ε are

$$I_{\text{eq}}(V) - I_{\text{vc}}(V) \approx I'_{\text{eq}}(V)\dot{V}, \frac{\tau_{1/2}}{\ln 2} \lesssim I'_{\text{eq}}(\widetilde{V})0.2\,\text{mV}, \tag{4.7}$$

where $\tau_{1/2}$ is the time for deviations from the transversally stable steady-state curve (4.6) to decay to half of their initial value. We estimate $\tau_{1/2}$ from recovery transients after disturbances naturally occuring from imperfections in the voltage clamp during VC runs as $\tau_{1/2} \lesssim 0.075\,\text{s}$, such that $\dot{V}\tau_{1/2}/\ln 2 \approx \dot{\widetilde{V}}\tau_{1/2}/\ln 2 \lesssim 0.2\,\text{mV}$. Thus, the systematic bias between $I_{\text{vc,sm}}(\widetilde{V})$ in Fig. 4.1(b) and the true steady state curve $I_{\text{eq}}(V)$ caused by dynamically changing $\widetilde{V}$ is below measurement disturbances.

Estimate (4.7) can also be tested *in silico*. Figure 4.4 emulates both VC and CC protocols with the Morris-Lecar model [6], which is of the form (4.3) with $j = 1$. For the chosen parameter set (see 4.2.3), the curves $(I_{\text{vc}}(t), \widetilde{V}(t))$ and $(I_{\text{eq}}(\widetilde{V}(t)), \widetilde{V}(t))$ are order ε ($\approx 1\%$) apart.

Consequently, time profile $(I_{\text{vc}}(t), \widetilde{V}(t))$ follows closely the curve (4.6) of *stable* steady states of the fast subsystem (4.3) with VC protocol $I_{\text{ext}} = g_{\text{c}}(V - \widetilde{V})$, treating $\widetilde{V}$ as a parameter. We now connect the curve (4.6) to a curve of fast-subsystem equilibria of the CC protocol, which is in part unstable. To this end we recast the VC protocol in the form of a CC protocol with non-constant current ramp speed $\varepsilon\Delta_{I,\text{vc}}(t)$ and disturbances: the smoothed time profile $I_{\text{vc,sm}}(\widetilde{V}(t))$ in Fig. 4.1(b)

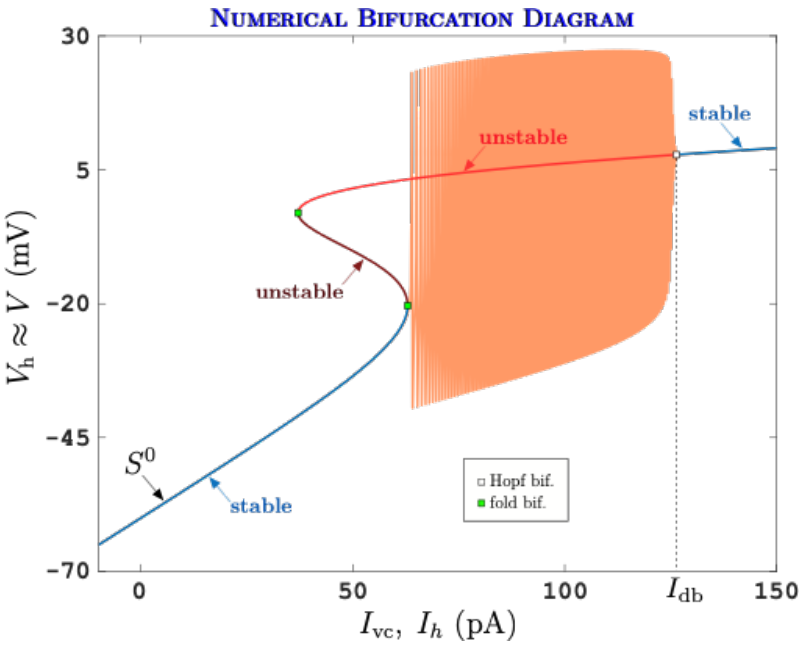

Figure 4.4: VC and CC *in silico*. Protocol as described for Fig. 4.1 applied to a type-I Morris-Lecar neuron model [6]; see 4.2.3 for parameter values. The two-dimensional fast subsystem has a S-shaped steady-state curve satisfying(4.6). The steady-state curve (4.6) and the (multi-color) S-shaped curve from the VC protocol indistinguishable throughout the range of input currents I_{vc}. The orange curve resulting from the CC protocol is very close to S^0 and the VC protocol near its dynamically stable parts.

of the VC run (thick, in blue/brown/red) equals the raw-data measured time profile $I_{vc}(t)$ (thin blue curve with fluctuations) plus disturbances $\eta_{vc}(t)$, defined by $\eta_{vc}(t) = I_{vc,sm}(\widetilde{V}(t)) - I_{vc}(t)$. After smoothing, the derivative $I'_{vc,sm}(\widetilde{V})$ w.r.t. $\widetilde{V}$ is moderate ($\lesssim 20\,\mathrm{pA/mV}$ in modulus at its maximum near I_{db}), such that the VC protocol implies

$$\begin{aligned} I_{ext} &\approx I_{vc,sm} + \eta_{vc}, \\ \dot{I}_{vc,sm} &= \varepsilon \Delta_{I,vc}(t), \end{aligned} \tag{4.8}$$

where $\Delta_{I,vc}(t) = I'_{vc,sm}(\widetilde{V}(t))\Delta_V \approx I'_{eq}(\widetilde{V}(t))\Delta_V$ with upper bound $\max_t |\Delta_{I,vc}(t)| \lesssim 3.7\,\mathrm{pA/ms}$ in the range of Fig. 4.1(b). Thus, $I_{vc,sm}$ is indeed still slow. Hence, except for disturbances $\eta_{vc}(t)$, the external current I_{ext} is slowly varying according to a CC protocol with slowly time-varying speed $\varepsilon\Delta_{I,vc}(t)$, such that the VC protocol (4.4) is equivalent CC protocol (4.8) with disturbances η_{vc}.

Systems (4.3), (4.5) and (4.3), (4.8) without disturbances ($\eta_{vc} = 0$) are both models of CC protocols. They have the same fast subsystem (4.3) when setting $\varepsilon = 0$ and identifying I_h and $I_{vc,sm}$. The respective fast-subsystem steady states $(V, x_j)_{j=1,\ldots,N}$ satisfy

$$\begin{aligned} \textstyle\sum_j I_j(x_j, V) &= I_h (= I_{vc,sm}), \\ x_j &= x_{\infty,j}(V). \end{aligned} \tag{4.9}$$

However, systems (4.3), (4.5) and (4.3), (4.8) differ by the nature of their respective slow variables I_h and $I_{vc,sm}$: I_h is an externally applied hold current for (4.5), while $I_{vc,sm}$ is a measured (and smoothed) current from the feedback control $g_c(V - \widetilde{V})$ of the voltage source for (4.8). Thus, while the S-shaped steady-state curve $(I_{eq}(V), V)$ is identical for both systems, it contains large unstable segments as a steady-state curve of (4.3), (4.5), while it always stable as a steady-state

curve of (4.3), (4.8). The change in stability is caused by the disturbances η_{vc}, which are current adjustments generated by the feedback term in (4.4), $I_{\mathrm{vc}} = g_{\mathrm{c}}(V - \widetilde{V})$. Along most of the curve $(I_{\mathrm{vc,sm}}(\widetilde{V}), \widetilde{V})$ the η_{vc} are small fluctuations such that $I_{\mathrm{vc,sm}}(\widetilde{V}) \approx I_{\mathrm{vc}}(\widetilde{V})$ and the feedback is approximately non-invasive [54]. Estimate (4.7) ensures that the measurements $I_{\mathrm{vc}}(V)$ stay close to $I_{\mathrm{eq}}(V)$. Therefore, we can conclude that the VC protocol (4.4) with slowly varying feedback reference signal $\widetilde{V}$ reveals the entire family of steady states of a neuron (type 1 or 2) with constant external current I_{ext}, both stable (observable) and unstable (non-observable, hidden). Consequently, the N-shaped I-V relations for type-1 neurons reported in [137, 138, 141] equal S-shaped steady-state bifurcation diagrams for these neurons with respect to I_{ext}. They are tractable with a VC protocol where the current I_{ext} is a sufficiently slowly varying feedback current with sufficiently small fluctuations η_{vc}. In particular, this allows us to detect and pass through fold bifurcations directly in the experiment.

In contrast, for the CC open-loop protocol (4.5) applying a slowly varying electrical current the neuron dynamically transitions between its observable rest states and its observable (dynamically stable) spiking states (see Fig. 4.1(b) orange timeprofile). The speed of variation $\varepsilon\Delta_I \lesssim 0.75 \times 10^{-3}\,\mathrm{pA/ms}$ is such that I_{ext} varies by only about 1 pA per spiking period. Transients to step current responses in the stable spiking region have a half-time for decay $\tau_{1/2} \lesssim 0.2\,\mathrm{s}$, permitting us to estimate the systematic bias between true steady-state spiking and the response to dynamically changing I_{ext}, similar to (4.7) for certain features of the periodic spike train. For example, the bias in the spike minimum V_{min} near $I_{\mathrm{h}} = 200\,\mathrm{mA}$ is $V'_{\mathrm{min}}(I_{\mathrm{h}}) \times \dot{I}_{\mathrm{h}} \times \tau_{1/2}/\ln 2 \sim 0.14\,\mathrm{mV}$ to first order of ε. Near the approximate Hopf bifurcation at $I_{\mathrm{h}} \approx I_{\mathrm{db}}$ in the experiment the bias will exceed order ε as both, $\tau_{1/2}$ and $V'_{\mathrm{min}}(I_{\mathrm{h}})$ approach infinity at Hopf bifurcations. Thus, combining VC protocol, recast as (4.8) varying $I_{\mathrm{vc,sm}}$, and the CC protocol (4.5), varying I_{h}, enables us to interpret the data sets from both protocols in Fig. 4.1(b) as a bifurcation diagram including unstable states.

For the experimental curves presented in Fig. 4.1(b,c) (and also in Fig. 4.2), the disturbances η_{vc} are not small in some unstable parts of the reported steady-state curve (e.g., near $\widetilde{V} = -30\,\mathrm{mV}$ in Fig. 4.1(b)), caused by imperfect voltage clamping across the membrane. Furthermore, the distances between $(I_{\mathrm{vc,sm}}(t), \widetilde{V}(t))$ in the VC run and $(I_{\mathrm{h}}(t), V_{\mathrm{cc}}(t))$ in the CC run for the same cell are visibly larger along parts of the curve corresponding to dynamically stable stationary states. This is due to the natural *drift* of the neuron's physiological properties as it changes dynamically from one protocol to the other.

4.5 Discussion

Tracking non-observable (hidden) states through their stability boundaries in experimental settings bridges the gap between real-world phenomena and nonlinear science. Specifically, closed-loop control methods with slow variations of feedback reference signals enable us to discover unstable underlying states of nonlinear systems. Future work will focus on employing fully *dynamic clamp* electrophysiology [59, 60, 62, 64], which involves two-way real-time communication between neuronal tissue and computer control, permitting non-invasive feedback control with complex reference signals. This will allow biologists to validate computational models by comparing their numerical bifurcation diagrams with experimental ones in parameter regions where the uncontrolled spiking response is dynamically unstable or sensitive with respect to system parameters.

This chapter is our first step in the study of bifurcation diagrams obtained throught an electrophysiological experiment. This indicates us that it is indeed possible to obtain approximate bifurcation diagram from neuron activity, and so giving us information on its excitability threshold, its unstable states and its excitability class. This also means that the bifurcation diagrams obtained from mathematical models are not just abstract, and can be obtained using the right tools from experiments.

Chapter 5

Control Based Continuation in Experiments (CBCE)

In the previous chapter we developed a technique for computing a bifurcation diagram from an experiment, which involves inaccessibility to the underlying equations of the system. Although this method has the advantage of being easy to apply, it has intermittent fluctuations around unstable steady states and is only an approximation. Furthermore, the use of the current clamp (CC) method obviously does not give access to the exact branches of periodic orbits. In this chapter, we propose to focus on another approach denoted *Control Based Continuation in Experiments* (CBCE) which can be seen as a refinement of an approach presented in Chapter 4, the CBCE has been developed for non-excitable systems, such as mechanical systems [54, 145]. Here we will apply CBCE in the context of excitable systems. The CBCE method makes it possible to obtain the exact branch of stationary points, but also the branch of stable and unstable periodic cycles, based on control theory [146, 147] and the a path-following approach [18, 38–40].

5.1 Introduction

CBCE provide a solution for the systematic exploration of complex nonlinear dynamic systems in the field of physical experiments. Just as numerical continuation methods follows solution branches of differential equations, control-based continuation techniques have emerged as a robust and versatile approach to studying the behavior of physical systems as key parameters vary [54, 145]. Control-based continuation, unlike traditional methods, allows to track these emerging solution branches and associated bifurcations in parameter spaces, thereby providing insight into regions of distinct dynamical regime. While numerical continuation methods excel when differential equations are available, the challenge arises when dealing with experimental setups. Control-based continuation provides a promising alternative, offering a more efficient and precise way to navigate through the complex landscape of physical experiences and uncover their underlying dynamic behaviors.

In Fig. 5.1 we try to give a conceptual intuition of CBCE: an experiment is in real-time progress, and we can interact with it only through the system parameters (as part of the continuation we will select a continuation parameter λ), but also through a control term described as

$$g_c(x,\lambda)(x-\widetilde{x}) \tag{5.1}$$

where $\widetilde{x}$ corresponds to the *reference signal* (or target) that we wish to reach for the continuation, x is the experiment output, and g_c the gain associated with this control. This signal $\widetilde{x}$ can be a

constant injected into the system in the case of continuation of stationary solutions from experiments, or periodic in the case of continuation of limit cycles from experiments. The term $(x-\widetilde{x})$ corresponds to what is called the *residual*: this difference makes it possible to evaluate how far the solution extracted from the experiment is from the reference signal. In the stationnary point case, the main idea behind this approach is to ensure that $\lim_{t\to\infty}|x(t)-\widetilde{x}|\to 0$. This signifies that the algorithm has successfully converged towards a solution of the uncontrolled problem. Similarly, in the case of periodic solutions, we want $\lim_{t\to\infty}F(\widetilde{x},\lambda,T)\in C_p([0,1],\mathbb{R}^n)$ where C_p is the space of periodic functions, and T, the period (See section 5.3.6). CBCE requires *closed-loop* protocol, because we want $\widetilde{x}$ to change iteratively.

The closed-loop has intrinsic delay τ. However, herein, we assume small delay $\tau\approx 0$. Hence $x(\widetilde{x},\lambda,\tau)\equiv x(\widetilde{x},\lambda,0)$. Moreover, the main assymption is that the experiment is controllable.

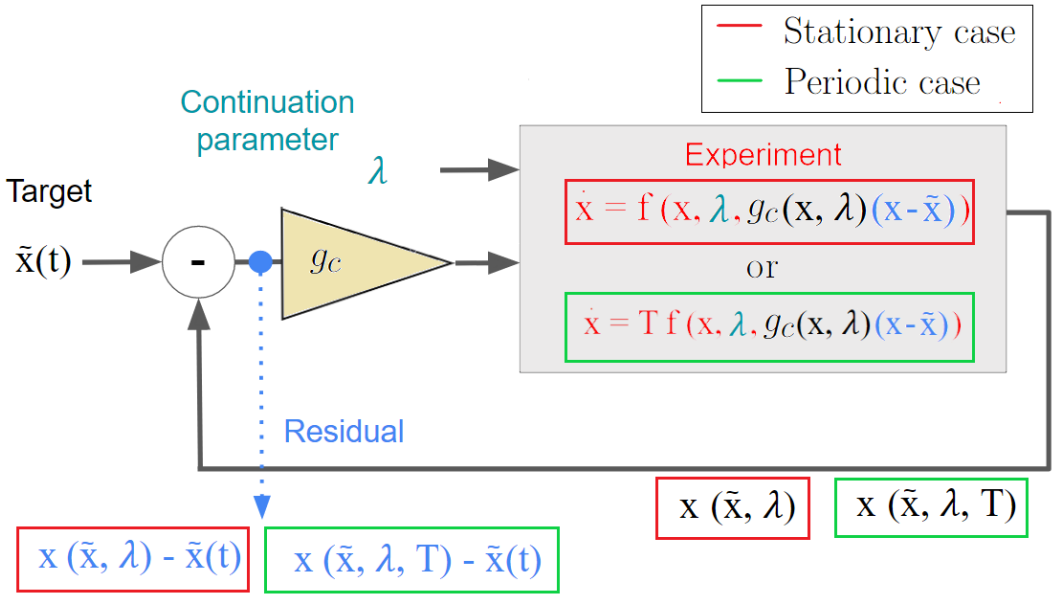

Figure 5.1: Feedback-control $g_c(x,\lambda)(x-\widetilde{x})$ applied to an experimental protocol. $\widetilde{x}(t)$ is the signal of reference, λ is the continuation parameter, g_c is the gain, the term in blue is the residual, and the equation in red represents a model of the controlled experiment.

For the periodic case [54, 145, 148, 149], we need to satisfy two primary assumptions before to proceed: 1) we assume that we have the ability to project our samples to a functional space, i.e. a Banach space $x(\widetilde{x},\lambda,t)\in C(\mathbb{R}^n)$, and 2) the residual $F(\widetilde{x},\lambda,t)$ converges. Thus, in this chapter, we will first give the different theoretical points necessary to understand the algorithm. Subsequently, we will explain this algorithm, first in the context of stationary points, then in the context of periodic solutions.

Moreover, although the output of the experiment is noisy and that the Lipschitz continuity condition is not satisfied, the CBCE method can be used to compute bifurcation diagram from the data, because all what is required is that the system is controllable and the residual converges.

5.2 Theoretical context

Before we proceed, I provide some computational algorithms used in the context of continuation.

5.2.1 Newton's method

Newton's method is useful for solving non linear equations and for finding the zeros (or roots) of non linear functions. Its basic principle is an iterative approximation of the solutions using the derivatives of the function. In other words, it constructs a sequence of iterates that quickly converge to the desired solution. The central idea behind Newton's method is to approximate the curve of the local function by a linear tangent at each step, and then to find the intersection of this tangent with the x-axis to obtain a better estimate of the solution. By repeating this iterative process, Newton's method converges to the solution with high accuracy, usually much faster than many other methods.

Algorithm: Let f be a differentiable map from U to $\mathbb{R}^n$, where U is an open subset of $\mathbb{R}^n$. Newton's method consists of starting with some guess a_0 for a solution of the nonlinear equation $f(x) = 0$. Then linearize the equation at a_0: replace the increment to the function $f(x) - f(a_0)$, by a linear function of the increment, $[Df(a_0)](x - a_0)$. Now solve the corresponding linear equation:

$$f(a_0) + Df(a_0)(x - a_0) = 0 \iff Df(a_0)(x - a_0) = -f(a_0) \tag{5.2}$$

If $Df(a_0)$ is invertible:

$$a_1 = a_0 - [Df(a_0)]^{-1} f(a_0) \tag{5.3}$$

Then we iterate this process to find the approximation of the location of the root of f. To stop the algorithm, we can determine a step limitation number or a criterion of accepting an approximate solution.

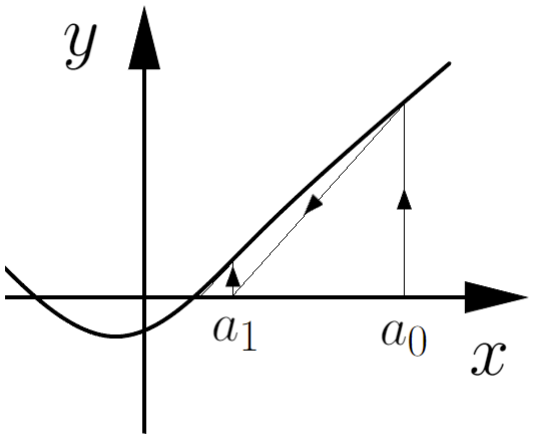

Figure 5.2: Newton's method principle: we start the process with an initial seed a_0, and we take the corresponding value on the curve. Then, we approximate the curve of the local function by a linear tangent t_0, then find the intersection of this tangent with the x-axis to obtain a better estimate of the solution, which will be a_1. We iterate the process until convergence.

5.2.2 Broyden's method

The Broyden's method, also known as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, is an important technique in numerical optimisation and the solution of systems of nonlinear equations. It was developed by Charles W. Broyden, Donald Goldfarb, David F. Shanno and Roger Fletcher in the 1960s and 1970s [150]. This method is particularly valuable for solving problems

where it is necessary to find a minimum (or maximum) of a nonlinear function, or to solve a system of nonlinear equations. Unlike Newton's method, Broyden's method does not require explicit calculation of the derivatives of the function, making it applicable to a wide variety of problems, even when the function is complex or expensive to evaluate.

Algorithm: Consider a system of k nonlinear equations:

$$f(x) = 0, \tag{5.4}$$

where f is a vector-valued function of vector x:

$$\begin{aligned} x &= (x_1, x_2, x_3, ..., x_k) \\ f(x) &= (f_1(x_1, x_2, x_3, ..., x_k), f_2(x_1, x_2, x_3, ..., x_k), ..., f_k(x_1, x_2, x_3, ..., x_k)) \end{aligned} \tag{5.5}$$

For such problems, Broyden's method gives a generalisation of the Newton method, replacing the derivative by the Jacobian $\mathbf{J}$. The Jacobian is determined iteratively, based on the secant equation in the finite difference approximation:

$$\mathbf{J}_n(x_n - x_{n-1}) \approx f(x_n) - f(x_{n-1}) \tag{5.6}$$

where n is the iteration index. For clarity, let us define:

$$\begin{aligned} f_n &= f(x_n) \\ \Delta x_n &= x_n - x_{n-1} \\ \Delta f_n &= f_n - f_{n-1} \end{aligned} \tag{5.7}$$

so the above may be rewritten as:

$$\mathbf{J}_n \Delta x_n \approx \Delta f_n \tag{5.8}$$

The above equation is underdetermined when k is greater than one. Broyden's method suggests using the current estimate of the Jacobian matrix J_{n-1} and improving upon it by taking the solution to the secant equation that is a minimal modification to J_{n-1}:

$$\mathbf{J}_n = \mathbf{J}_{n-1} + \frac{\Delta f_n - \mathbf{J}_{n-1}\Delta x_n}{||\Delta x_n||^2} \Delta x_n^T \tag{5.9}$$

We may then proceed with the Newton iteration:

$$x_{n+1} = x_n - \mathbf{J}_n^{-1} f(x_n) \tag{5.10}$$

5.2.3 Spectral decomposition by normalized Fourier series

The Fourier series is a spectral method that exploits the representation in terms of frequencies to approximate periodic functions. It is part of the broader family of *spectral methods*, which encompasses different approaches based on the analysis of the spectral properties of operators or functions.

Suppose an experiment has a periodic output $x(t+T)=x(t)$ for all t, and where $T\in\mathbb{R}$ is the period. Moreover, $\widetilde{x}(t+T)=\widetilde{x}(t)$. We consider a Fourier representation:

$$\Pi_q : C_p([0,1];\mathbb{R}^n) \to \mathbb{R}^{(2q+1)n}, \tag{5.11}$$

where Π_q is the projection operator, $(2q+1)$ is the number of Fourier coefficients, and n, the dimension in which the signal x and $\widetilde{x}$ lives. In this chapter, we will only study the voltage output of our experiment so $n=1$. So, if we take a signal $\widetilde{x}(t)$, and we fix a q value, we will have: $\Pi_q[\widetilde{x}(t)]=[\widetilde{x}_{-q},...,\widetilde{x}_{-1},\widetilde{x}_0,\widetilde{x}_1,...,\widetilde{x}_q]$. Moreover, for a Fourier function, we know how to compute these coefficients:

$$\begin{aligned}
\widetilde{x}_{-j} &= \frac{2}{T}\int_0^T \widetilde{x}(t)\cos(j\omega t)dt \\
\widetilde{x}_0 &= \frac{1}{T}\int_0^T \widetilde{x}(t)dt \\
\widetilde{x}_j &= \frac{2}{T}\int_0^T \widetilde{x}(t)(t)\sin(j\omega t)dt
\end{aligned} \tag{5.12}$$

We will use normalized Fourier coefficients. To do so, we just have to introduce a normalization constant $\frac{1}{\sqrt{2\pi}}$ (which will not affect the frequency, but only the amplitude size), such that we obtain:

$$\begin{aligned}
\widetilde{x}_{-j} &= \int_0^T \widetilde{x}(t)\sqrt{\frac{2}{T}}\cos(2\pi jt)dt, \\
\widetilde{x}_0 &= \int_0^T \widetilde{x}(t)\sqrt{\frac{1}{T}}dt, \\
\widetilde{x}_j &= \int_0^T \widetilde{x}(t)\sqrt{\frac{2}{T}}\sin(2\pi jt)dt,
\end{aligned} \tag{5.13}$$

Then the Fourier series associated with these normalized coefficients is:

$$\widetilde{x}(t) = \widetilde{x}_0\sqrt{\frac{1}{T}} + \sum_{j=1}^{q}[\widetilde{x}_{-j}\sqrt{\frac{2}{T}}\cos(2j\pi t) + \widetilde{x}_j\sqrt{\frac{2}{T}}\sin(2j\pi t)] \tag{5.14}$$

5.3 Algorithmes and simulated experiments

Here, we introduce our main test case for CBCE, namely the FitzHugh-Nagumo (FHN) model. We first analyse it analytically and with numerical continuation. Finally, we proceed with CBCE.

5.3.1 Modified FitzHugh Nagumo model

The FHN model [4, 5] is a nonlinear system of differential equations describing a prototypical neuron. It is a simplified 2D version of the Hodgkin-Huxley model, which captures in a detailed way both the electrical activity of the membrane potential (via Kirchoff's law) and the dynamics of the ion channels of a spiking neuron.

The classical version of FHN [151] is defined as follows:

$$\begin{aligned} \dot{V} &= V - \frac{V^3}{3} - W + I_{\text{ext}}, \\ \dot{W} &= \varepsilon(V + \alpha - \beta W), \end{aligned} \tag{5.15}$$

where V is the membrane potential, W is the linear recovery variable (mimicking channel dynamics), and I_{ext} is an external current applied to the neuron. Parameters α and β are tuneable and control the position of the linear nullcline. Finally, $0 < \varepsilon \ll 1$ is a small parameter that controls the timescale separation between the two state variables.

Here, we will use a modified version of the system (5.15) on which the construction of an equivalent analog circuit was based; see [152]. Namely, the modified FHN equations read:

$$\begin{aligned} \dot{V} &= -V(V-a)(V-b) - W + I_{ext}, \\ \dot{W} &= \varepsilon(cV + dW + e), \end{aligned} \tag{5.16}$$

where V and W still denote membrane potential and linear recovery, respectively. Parameters a, b, c, d and e are tuneable. Note that, contrary to the version studied in [152], we retain the possibility to control the angle of the linear nullcline (as in the original FHN model). System (5.16) has a cubic V-nullcline and a linear W-nullcline, of equations:

$$\begin{aligned} W &= -V(V-a)(V-b) + I_{ext}, \\ W &= -\frac{cV+e}{d}, \end{aligned} \tag{5.17}$$

respectively, from which we can obtain the coordinates of the equilibria. The stability of any of these equilibria, (V^*, W^*) can be assessed through the Jacobian matrix $\mathbf{J}$, which takes the form:

$$\mathbf{J} = \begin{pmatrix} -3V^{*2} + 2(a+b)V^* - ab & -1 \\ \varepsilon c & \varepsilon d \end{pmatrix} \tag{5.18}$$

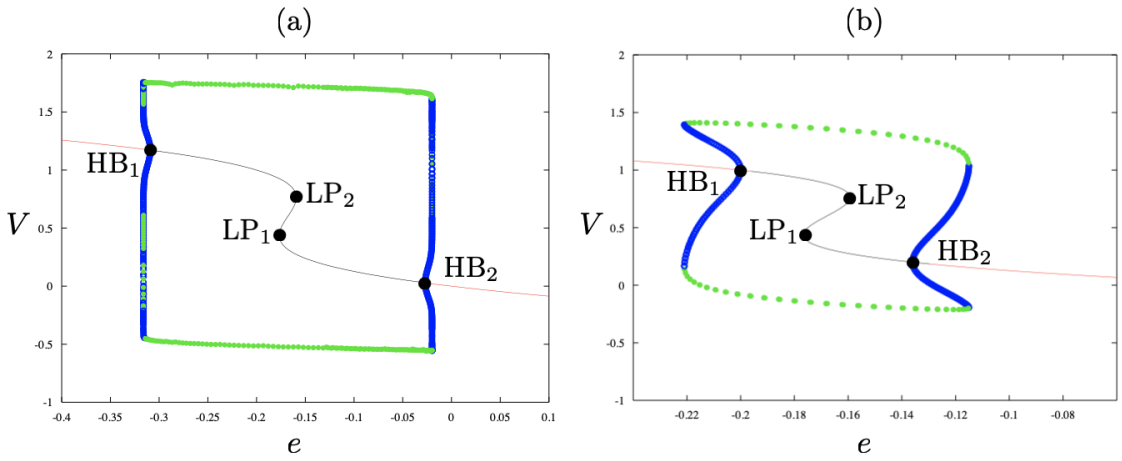

Figure 5.3: (a) Bifurcation diagram of the modified FHN model (5.16) with respect to parameter e: stable (resp. unstable) families of equilibria are represented by red (resp. black) lines; stable (resp. unstable) families of limit cycles are represented in green (resp. blue). Parameter values are: $a = 1.8$, $b = 0$, $c = 1$, $d = -1$, $I_{\text{ext}} = 0$, $\varepsilon = 0.1$. (b) Same as (a) but with $\varepsilon = 0.6$.

We now focus on the bifurcation diagram of the modified FHN model (5.16) with respect to

parameter e. The bifurcation diagram shown in Fig 5.3 (a) display two subcritical Hopf bifurcations (denoted HB_1 and HB_2, respectively). From each of them, a branch unstable limit cycles emerges very "rapidly", in terms of parameter variations. These families of limit cycles correspond to canard solutions [34, 72–74], and the fact that they only exist in very narrow intervals of parameter values implies that one usually refers to these branches as canard explosions. The system also displays two saddle-node bifurcations, marked as $LP_{1,2}$ (for Limit Point), as the parameter is varied.

In Fig 5.3 (b), we have increased the value of the timescale parameter ε, which effectively makes the explosive behaviour of the branches of limit cycles disappear. It nevertheless maintains the criticality of the Hopf bifurcations. In this case, the FHN model behaves locally much more like a standard subcritical Hopf normal form, and it should be easier to compute the associated bifurcation diagram once in the experimental context.

Adjusting parameters values in system (5.16), one can also change the criticality of both Hopf bifurcations, and obtain a supercritical scenario on each side, as showcased in Fig. 5.4.

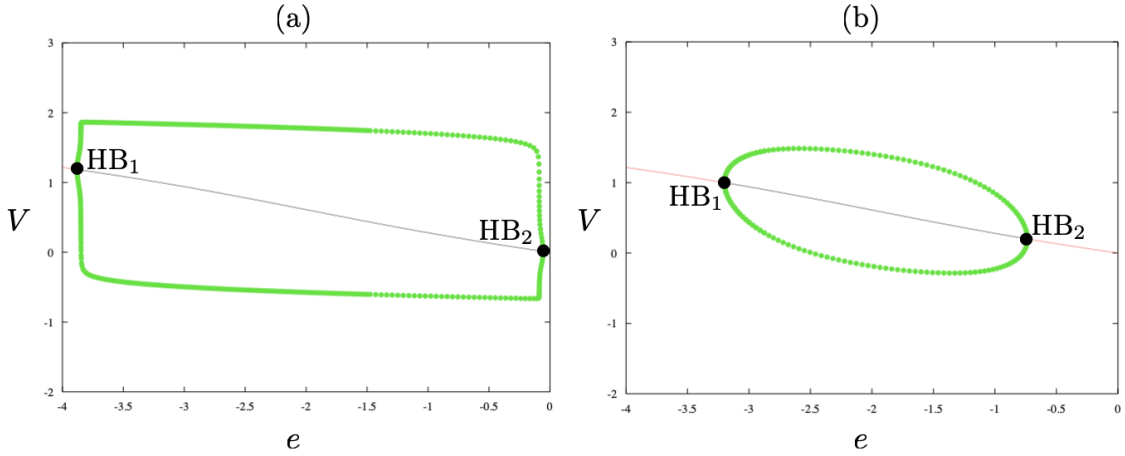

Figure 5.4: (a) Supercritical Hopf scenario in the bifurcation diagram of system (5.16) with respect to e. Parameter values are: $a = 1.8$, $b = 0$, $c = 4$, $d = -1$, $I_{\text{ext}} = 0$, $\varepsilon = 0.05$, which induces a canard explosion near each Hopf bifurcation. (b) Same as (a) but with a timescale separation due to the parameter value $\varepsilon = 0.6$. The branches of limit cycles are not explosive because parameter ε is too large.

Now, we want to introduce a feedback control term in the V equation of system (5.16), which will enable us to predict the behaviour of the associated analog circuit in closed-loop control model. The modified equations then read:

$$\begin{aligned} \dot{V} &= -V(V-a)(V-b) - W + g_c(V-\widetilde{V}), \\ \dot{W} &= \varepsilon(cV + dW + e), \end{aligned} \tag{5.19}$$

where g_c is the control gain, which allows to correct the inputs, and $\widetilde{V}$ is the reference signal. So the term $g_c(V-\widetilde{V})$ replaces the external current I_{ext}. Then, we can determine the appropriate range of g_c values in order to stabilise system (5.19). The new Jacobian matrix at an equilibrium point (V^{ast}, W^{ast}) is:

$$\mathbf{J} = \begin{pmatrix} -3V^{*2} + 2(a+b)V^* - ab + g_c & -1 \\ \varepsilon c & \varepsilon d \end{pmatrix} \tag{5.20}$$

The zeros of the trace of $\mathbf{J}$ correspond to Hopf bifurcation points. For a large range of g_c values, there are two Hopf points when varying one of the parameters, say e. The condition for

such Hopf points to occur is given by:

$$\mathrm{tr}(\mathbf{J}) := -3V^{*2} + 2(a+b)V^* - ab + g_c + \varepsilon d = 0. \tag{5.21}$$

However, there is a range of g_c values for which there are no Hopf bifurcations (and no saddle-node bifurcations either), hence giving only one stable equilibrium for all values of the primary parameter. The critical g_c value marks the fact that both Hopf points coincide at degenerate Hopf bifurcation, before disappearing, and that corresponds to when the roots of eq. (5.21) coincide, namely when

$$4(a+b)^2 = 12(ab - g_c - \varepsilon d),$$

which can be rearranged into:

$$g_{c,\mathrm{dh}} = ab - \varepsilon d - \frac{1}{3}(a+b)^2. \tag{5.22}$$

On the other hand, system (5.19) can also have saddle-node bifurcations, which will induce instabilities. The condition for such bifurcations is given by the zero of the determinant of $\mathbf{J}$, which gives the following equation:

$$\det(\mathbf{J}) := \varepsilon d \left(-3V^{*2} + 2(a+b)V^* - ab + g_c\right) + \varepsilon c = 0. \tag{5.23}$$

The roots of eq. (5.23) coincide at a cusp bifurcation, that is, when

$$4\varepsilon^2 d^2 (a+b)^2 = 12\varepsilon d \left(\varepsilon d(ab - g_c) - \varepsilon c\right),$$

which simplifies to

$$d(a+b)^2 = 3\left(d(ab - g_c) - c\right).$$

This gives another specific value of g_c, namely:

$$g_{c,\mathrm{cusp}} = ab - \frac{c}{d} - \frac{1}{3}(a+b)^2. \tag{5.24}$$

For the parameter values that we have chosen (see Figs. 5.3 and 5.4), it turns out that $g_{c,\mathrm{dh}} < g_{c,\mathrm{cusp}}$. Therefore, for $g_c < g_{c,\mathrm{dh}}$, the controlled system (5.19) is always stable.

5.3.2 CBCE algorithm for the stationary case

Step 1: Initialization step.

• Choose g_c (the gain of the control) such that it stabilize the system. So we assume $u = g_c(x - \widetilde{x})$ where u is the control and $(x - \widetilde{x})$ is the residual for the control. Choose a small parameter for the finite-difference approximation of the Jacobian matrix $h \in \mathbb{R}$, a small continuation stepsize $\Delta s \in \mathbb{R}$.

• Suppose we have an initial guess: $y_0 = (\widetilde{x}_0, \lambda_0)$ with $\tilde{x}_0 \in \mathbb{R}^n$, $\lambda_0 \in \mathbb{R}$, so $y_0 \in \mathbb{R}^{n+1}$.

• Choose arbitrarily the direction vector: $\tau_0 = (\dot{\widetilde{x}}_0, \dot{\lambda}_0) = (0, 1)$

Step 1 end.

Step 2: Prediction step.

• Compute the next point $y_1 = (\tilde{x}_1, \lambda_1)$ by applying a predictor step (Tangent predictor):

$$\begin{bmatrix} \widetilde{x}_{\text{pred}} \\ \lambda_{\text{pred}} \end{bmatrix} = \begin{bmatrix} \widetilde{x}_0 \\ \lambda_0 \end{bmatrix} + \Delta s \begin{bmatrix} \dot{\widetilde{x}}_0 \\ \dot{\lambda}_0 \end{bmatrix} \tag{5.25}$$

Step 2 end.

Step 3: Correction step

• Run experiment with $(\widetilde{x}_1^{(k)}, \lambda_1^{(k)})$, where k is the Newton's correction iteration, read the data x_i with $i = 1, ..., n.$, and determine the asymptotic regime x_∞, usually, for equilibria, the average of the final x_i.

• Evaluate the residual $F(\widetilde{x}_1^{(k)}, \lambda_1^{(k)}) = x(\widetilde{x}_1^{(k)}, \lambda_1^{(k)}) - \widetilde{x}_{\text{pred}}$.

• Construct the extended system:

$$H(y; \Delta s) = \begin{bmatrix} F_{\text{ref}} \\ (\widetilde{x} - \widetilde{x}_0)\dot{\widetilde{x}}_0 + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s \end{bmatrix} \tag{5.26}$$

And apply the Newton's correction step:

$$\begin{bmatrix} \widetilde{x}_1^{(k+1)} \\ \lambda_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \widetilde{x}_1^{(k)} \\ \lambda_1^{(k)} \end{bmatrix} - \begin{bmatrix} F_x^{(k)} & F_\lambda^{(k)} \\ \dot{\widetilde{x}}_0 & \dot{\lambda}_0 \end{bmatrix}^{-1} \begin{bmatrix} F_{\text{ref}} \\ (\widetilde{x}_1^{(k)} - \widetilde{x}_0)\dot{\widetilde{x}}_0 + (\lambda_1^{(k)} - \lambda_0)\dot{\lambda}_0 - \Delta s \end{bmatrix} \tag{5.27}$$

When $||H_k(y; \Delta s)|| < tol$ and when $|| \begin{bmatrix} \widetilde{x}_1^{(k)} \\ \lambda_1^{(k)} \end{bmatrix} - \begin{bmatrix} \widetilde{x}_1^{(k-1)} \\ \lambda_1^{(k-1)} \end{bmatrix} || < tol$,

allocate $(\widetilde{x}_1, \lambda_1) = (\widetilde{x}_1^{(k_{\text{final}})}, \lambda_1^{(k_{\text{final}})})$. (Convergence)

Step 3 end.

Step 4: direction vector update step.

• Normalize the new direction vector.

• Evaluate next tangent direction:

$$\begin{bmatrix} F_{\widetilde{x}_1} & F_{\lambda_1} \\ \dot{\widetilde{x}}_0 & \dot{\lambda}_0 \end{bmatrix} \begin{bmatrix} \dot{\widetilde{x}}_1 \\ \dot{\lambda}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.28}$$

• Rescale afterward the new tangent vector to length 1.

Step 4 end.

5.3.3 Broyden's Jacobian matrix calculation variant

The computation of the Jacobian matrix of our particular system is quite computationally and time consuming. In fact, in order to have a complete Jacobian, we have to carry out three experiments

to compute three residuals, and thus the two derivatives $F_x^{(k)}$ and $F_\lambda^{(k)}$, for each Newton iteration. To save time for the experiments, we also propose to replace this way of computing the Jacobian by Broyden's method described below.

We want to make a continuation with respect to only one parameter λ, and we consider a small perturbation h:

Step 3: Broyden's correction step.

If Broyden's iteration index k equals 1:

• Run experiment with $(\widetilde{x}_1^{(k)}, \lambda_1^{(k)})$, read the data $x_{\text{ref},i}$ with $i = 1,...,n.$, and determine $x_{\text{ref},\infty}$, usually, for equilibria, the average of the final $x_{\text{ref},i}$.

• Evaluate the residual $F_{\text{ref}}(\widetilde{x}_1^{(k)}, \lambda_1^{(k)}) = x_{\text{ref},\infty} - \widetilde{x}_{\text{pred}}$.

• Do the same with $(\widetilde{x}_1^{(k)} + h, \lambda_1^{(k)})$ and $(\widetilde{x}_1^{(k)}, \lambda_1^{(k)} + h)$, and evaluate the residuals $F_{\widetilde{x}\text{pert}}(\widetilde{x}_1^{(k)} + h, \lambda_1^{(k)}) = x_{\widetilde{x}\text{pert},\infty} - (\widetilde{x}_{\text{pred}} + h)$ and $F_{\lambda\text{pert}}(\widetilde{x}_1^{(k)}, \lambda_1^{(k)} + h) = x_{\lambda\text{pert},\infty} - \widetilde{x}_{\text{pred}}$.

• Compute the Jacobian matrix:

$$J_k = \begin{bmatrix} F_x^{(k)} = \frac{1}{h}(F_{\widetilde{x}\text{pert}} - F_{\text{ref}}) & F_\lambda^{(k)} = \frac{1}{h}(F_{\lambda\text{pert}} - F_{\text{ref}}) \\ \dot{\widetilde{x}}_0 & \dot{\lambda}_0 \end{bmatrix} \tag{5.29}$$

• Compute the extended problem:

$$H_k(y;\Delta s) = \begin{bmatrix} F_{\text{ref}} \\ (\widetilde{x} - \widetilde{x}_0)\dot{\widetilde{x}}_0 + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s \end{bmatrix} \tag{5.30}$$

Else:

• Run experiment with $(\widetilde{x}_1^{(k)}, \lambda_1^{(k)})$, read the data $x_{\text{ref},i}$ with $i = 1,...,n.$, and determine $x_{\text{ref},\infty}$, usually, for equilibria, the average of the final $x_{\text{ref},i}$.

• Evaluate the residual $F_{\text{ref}}(\widetilde{x}_1^{(k)}, \lambda_1^{(k)}) = x_{\text{ref},\infty} - \widetilde{x}_{\text{pred}}$.

• Compute the extended problem:

$$H_k(y;\Delta s) = \begin{bmatrix} F_{\text{ref}} \\ (\widetilde{x} - \widetilde{x}_0)\dot{\widetilde{x}}_0 + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s \end{bmatrix} \tag{5.31}$$

• Compute the Jacobian matrix:

$$\mathbf{J}_k = \mathbf{J}_{k-1} + \frac{\Delta f_k - \mathbf{J}_{k-1}\Delta x_k}{||\Delta x_k||^2}\Delta x_k^T \tag{5.32}$$

where:

$$\begin{aligned} \Delta x_k &= \begin{bmatrix} \widetilde{x}_1^{(k)} \\ \lambda_1^{(k)} \end{bmatrix} - \begin{bmatrix} \widetilde{x}_1^{(k-1)} \\ \lambda_1^{(k-1)} \end{bmatrix} \\ \Delta f_k &= H_k(y;\Delta s) - H_{k-1}(y;\Delta s) \end{aligned} \tag{5.33}$$

Step 3 end.

5.3.4 Secant direction update variant

In step 4 of the CBCE algorithm, we normally calculate the tangent vector of the last computed point. Because of the noise, even if we compute the average of the asymptotic section of the time series, the continuation curve may deviate from the correct path. To correct this, we try to calculate the secant vector as a direction vector as follows:

Step 4: Secant direction update step.

• Evaluate next secant direction:

$$\begin{bmatrix} \dot{\widetilde{x}}_1 \\ \dot{\lambda}_1 \end{bmatrix} = \begin{bmatrix} \widetilde{x}_1 \\ \lambda_1 \end{bmatrix} - \begin{bmatrix} \widetilde{x}_0 \\ \lambda_0 \end{bmatrix} \tag{5.34}$$

• Rescale afterward the new secant vector to length 1.

Step 4 end.

5.3.5 Trust region and multiple trials

In order to improve the code so that it can produce viable bifurcation diagrams in the presence of noise, we need to make some adjustments. First, we want to prevent our prediction from jumping too far due to noise. To do this, we have added the notion of *trust region*, which allows us to set a circular zone around the last computed point. If the calculated point is in the trust region, it is kept, otherwise it is recalculated. The trust region is defined as follows

$$(V_p^{(k)} - V_{p-1})^2 + (\lambda_p^{(k)} - \lambda_{p-1})^2 < R^2 + \varepsilon_{tol}, \tag{5.35}$$

where R is the distance between the last two computed points (V_{p-1}, λ_{p-1}) and (V_{p-2}, λ_{p-2}) (for the first point to be computed, we assume a large enough radius for the first guess in order to reach the first prediction correctly), and ε_{tol} is a small parameter allowing some distance variation between the computed points.

In addition, to refine our diagrams, we have added the ability to calculate multiple trials of a new point and take the average of the new coordinates for our bifurcation diagram. In this way, if the calculated point is in the trust region but slightly off the correct direction, the set of calculated points will allow you to keep the correct heading. The only problem with this method is, of course, the trade-off between the number of trials we want to compute and the computation time. In fact, in the case of high noise, it would be necessary to calculate a large number of points to obtain a good average of the coordinates, but the calculation would be very long. For our case, between 3 and 5 trials per point seems to be a reasonable ratio.

5.3.6 Periodic solution case

In this section 5.3.6, we will discuss how we calculate a periodic branch of a bifurcation diagram. The algorithm for the periodic case is very similar to the fixed points case but has some additional considerations. Suppose an experiment has a periodic output $x(t+T) = x(t)$ for all t, and where $T \in \mathbb{R}$ is the periodic output. During the periodic orbits branch continuation, because each solution can have different periods, we have to determine T as an unknown in our Newton's method. To approximate this periodic output, we will use a category of methods called *spectral methods*. This class of representative functions will allow us to represent a periodic solution as a linear combination of known functions, defined over the entire domain of study. The coefficients of this linear combination are obtained by projection, and to compute them, we want first so to rescale the differential equation in such a way that T becomes an explicit parameter:

$$\begin{aligned} \tau = \frac{t}{T} &\Longleftrightarrow t(\tau) = \tau T \\ &\Longrightarrow \frac{dx}{d\tau} = \frac{dx}{dt}\frac{dt}{d\tau} \\ &\Longrightarrow \quad \frac{dx}{d\tau} = T f(x, \lambda) \end{aligned} \tag{5.36}$$

By expressing explicitely T, the equation $\dot{x}$ is no longer defined between 0 and T, but 0 and 1, thus our unknowns (uncluding T) will be computed for this new rescaled period. In this chapter, we will use the spectral decomposition by Fourier series which allows us to represent a periodic function efficiently using a finite number of sine and cosine terms. Thus the space generated by the Fourier basis of periodic functions of period equals to 1 truncated at $Q = 2q+1$ Fourier modes is:

$$\Pi_Q = \{1, \cos(\omega t), \sin(\omega t), \cos(2\omega t), \sin(2\omega t), \cdots, \cos(Q\omega t), \sin(Q\omega t)\}$$

provided with the scalar product:

$$< u, v >= \int_0^1 u(t)v(t)dt$$

We then seek an approximation of the solution x of the form of its truncated Fourier series with $2q+1$ modes:

$$\hat{x}(t) = u_0 + \sum_{k=1}^{Q} \left[x_{2k-1} \cos(k\omega t) + x_{2k} \sin(k\omega t) \right].$$

Now, we have so $2q+3$ unknowns to resolve: the $2q+1$ residuals coming from the $2q+1$ Fourier modes, the period T, and the parameter λ. The construction of the extended problem and the Jacobian matrix for Newton's method are so different from the fixed point case:

$$H(y,s) = \begin{bmatrix} \text{Residuals } (2q+1 \text{ equations}) \\ \text{Phase condition (1 equation)} \\ \text{Pseudo-arclength condition (1 equation)} \end{bmatrix} \tag{5.37}$$

The residuals will take the same form as the residuals in the stationary case, that is to say $\Pi_q[F(\widetilde{x}, \lambda, T)] = \Pi_q[x(\widetilde{x}, \lambda, T)] - \Pi_q[\widetilde{x}]$, where Π_q is the Fourier decomposition into coefficients with q modes. In the context $q = 1$ modes, we have: $\Pi_1[F] = \{F_{-1}, F_0, F_1\}$.

The phase condition specifies how successive cycles of the solution align with each other. A disturbance or change in the parameters can result in a change in the phase condition, and it is essential to take this into account to ensure the stability of the continuation. Suppose we have a previous periodic solution, called $\widetilde{x}_{i-1}$. To avoid that the next computed solution is a rotated version of $\widetilde{x}_{i-1}$, we use a phase condition. Hence for example:

$$[\widetilde{x}_i(0) - \widetilde{x}_{i-1}(0)] \cdot \dot{\widetilde{x}}_{i-1}(0) = 0 \tag{5.38}$$

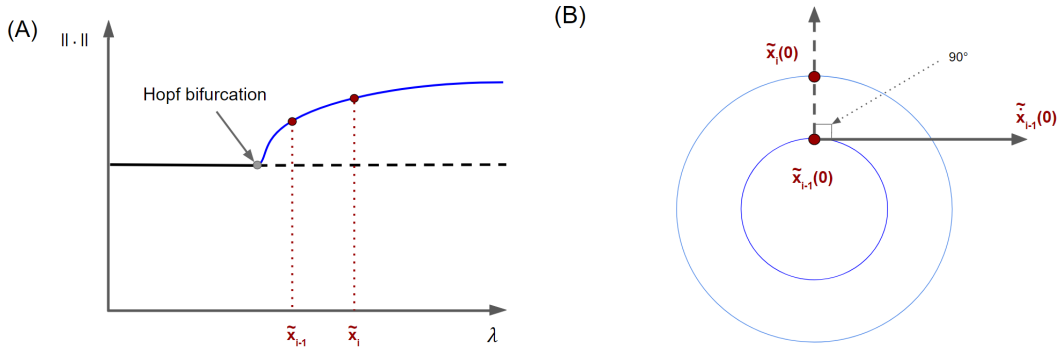

Figure 5.5: (A) Sketched bifurcation diagram. The black (solid and dotted) lines are the fixed points and the blue curve is the periodic solutions. We are taking two periodic solutions $\widetilde{x}_{i-1}$ and $\widetilde{x}_i$. (B) Sketch of the phase condition. The two periodic orbits are represented in blue, and one point is selected on the periodic orbit $\widetilde{x}_{i-1}$ and we want to force this point to not rotate on the next solution $\widetilde{x}_i$. To do so, we compute the tangent vector of this point and we want this vector to be perpendicular to the vector $\widetilde{x}_i - \widetilde{x}_{i-1}$.

We implement an integral version of the phase condition 5.38 anamely:

$$\frac{1}{T}\int_0^T [\widetilde{x}_i(t+\sigma) - \widetilde{x}_{i-1}(t)] \cdot \dot{\widetilde{x}}_i(t+\sigma)dt = 0 \tag{5.39}$$

for every σ. Writing $\widetilde{x}_i(t) \equiv \widetilde{x}_i(t+\sigma)$, gives:

$$\begin{aligned} &\frac{1}{T_{i-1}}\int_0^{T_{i-1}} [\widetilde{x}_i(t) - \widetilde{x}_{i-1}(t)] \cdot \dot{\widetilde{x}}_i(t)dt = 0 \\ \Leftrightarrow &\frac{1}{T_{i-1}}\int_0^{T_{i-1}} \widetilde{x}_i(t)^T \dot{\widetilde{x}}_{i-1}(t)dt = 0 \end{aligned} \tag{5.40}$$

Where $\widetilde{x}_i(t)^T$ is a transposed vector. This equation fixes the phase of successive computed periodic orbits. We discretise this condition:

$$\Leftrightarrow \sum_{j=-1}^{-q} \widetilde{x}_{j,i} \cdot \dot{\widetilde{x}}_{j,i-1} - \sum_{j=0}^{q} \widetilde{x}_{j,i} \cdot \dot{\widetilde{x}}_{j,i-1} \tag{5.41}$$

So in the q = 1 case, we have:

$$\Rightarrow \widetilde{x}_{0,i} \cdot \dot{\widetilde{x}}_{0,i-1} + \widetilde{x}_{-1,i} \cdot \dot{\widetilde{x}}_{-1,i-1} - \widetilde{x}_{1,i} \cdot \dot{\widetilde{x}}_{1,i-1} \tag{5.42}$$

The last equation of the extended problem, is the pseudo-arclength continuation rule:

$$\frac{1}{T_{i-1}}\int_0^{T_{i-1}} (\widetilde{x}_i^{(k)}(t) - \widetilde{x}_{i-1}(t)) \cdot \dot{x}_{i-1}(t) + (T_i^{(k)} - T_{i-1}) \cdot \dot{T}_{i-1} + (\lambda_i^{(k)} - \lambda_{i-1}) \cdot \dot{\lambda}_{i-1} - \Delta s \tag{5.43}$$

This new version of the equation takes into consideration the fact that we study Fourier coefficients of the periodic signal, but also the period of this solution. Here, we obtain the general formulation of the extended problem:

$$H(y,\Delta s) = \begin{bmatrix} \Pi_q[F(\widetilde{x},\lambda,T)] \\ \frac{1}{T}\int_0^T \widetilde{x}_i(t)^T \cdot \dot{\widetilde{x}}_{i-1}(t)dt \\ \frac{\int_0^{T_{i-1}} (\widetilde{x}_i^{(k)}(t)-\widetilde{x}_{i-1}(t))\cdot \dot{x}_{i-1}(t)dt}{T_{i-1}} + (T_i^{(k)}-T_{i-1})\cdot \dot{T}_{i-1} + (\lambda_i^{(k)}-\lambda_{i-1})\cdot \dot{\lambda}_{i-1} - \Delta s \end{bmatrix} \\ = 0 \tag{5.44}$$

5.3.7 CBCE algorithm for the periodic case

Step 1: Initialization step.

- Choose q, namely, the number of Fourier modes to approximate the periodic solution.
- Choose g_c (the gain of the control) such that it stabilize the system.
- Choose a small parameter for the finite-difference approximation of the Jacobian matrix $h \in \mathbb{R}$, a small continuation stepsize $\Delta s \in \mathbb{R}$.
- Suppose we have an initial guess: $y_0 = (\widetilde{x}_0, T_0, \lambda_0)$ with $\tilde{x}_0 \in \mathbb{R}^{n+1}$, $T_0 \in \mathbb{R}$, $\lambda_0 \in \mathbb{R}$, so $y_0 \in \mathbb{R}^{2q+3}$.
- Choose arbitrarily the direction vector: $\tau_0 = (\dot{\widetilde{x}}_0, \dot{T}_0, \dot{\lambda}_0) = (0,1)$

Step 1 end.

Step 2: Prediction step.

- Compute the next point $y_1 = (\widetilde{x}_1, T_1, \lambda_1)$ by applying a predictor step (Tangent predictor):

$$\begin{bmatrix} \widetilde{x}_{\text{pred}} \\ T_{\text{pred}} \\ \lambda_{\text{pred}} \end{bmatrix} = \begin{bmatrix} \widetilde{x}_0 \\ T_0 \\ \lambda_0 \end{bmatrix} + \Delta s \begin{bmatrix} \dot{\widetilde{x}}_0 \\ \dot{T}_0 \\ \dot{\lambda}_0 \end{bmatrix} \tag{5.45}$$

Step 2 end.

Step 3: Correction step

- Run experiment with $(\widetilde{x}_1^{(k)}, T_1^{(k)}, \lambda_1^{(k)})$, where k is the Newton's correction iteration, read the data, and extract a period x_i of the signal, with $i = 1, ..., T$.
- Apply the spectral decomposition by normalized Fourier coefficients on x_i to obtain $x(\widetilde{x}_1^{(k)}, T_1^{(k)}, \lambda_1^{(k)})$.
- Evaluate the residual $F(\widetilde{x}, T, \lambda) = x(\widetilde{x}_1^{(k)}, T_1^{(k)}, \lambda_1^{(k)}) - \widetilde{x}_{\text{pred}}$.
- Run experiment with $(\widetilde{x}_0, T_0, \lambda_0)$, read the data, and extract a period of the signal. Compute this derivative with respect to the time $x_{\Delta t}(\widetilde{x}_0^{(k)}, T_0^{(k)}, \lambda_0^{(k)})$, and apply the spectral decomposition by normalized Fourier coefficients on this signal in order to obtain $\Pi_q[x_{\Delta t}(\widetilde{x}_0, T_0, \lambda_0)]$.
- Construct the extended system:

$$H(y,\Delta s) = \begin{bmatrix} \Pi_q[F(\widetilde{x},T,\lambda)] \\ \frac{1}{T}\int_0^T \widetilde{x}_i(t)^T \cdot \dot{\widetilde{x}}_{i-1}(t)dt \\ \frac{1}{T_{i-1}}\int_0^{T_{i-1}} (\widetilde{x}_i^{(k)}(t) - \widetilde{x}_{i-1}(t))\cdot \dot{x}_{i-1}(t)dt ... \\ ... + (T_i^{(k)} - T_{i-1})\cdot \dot{T}_{i-1} + (\lambda_i^{(k)} - \lambda_{i-1})\cdot \dot{\lambda}_{i-1} - \Delta s \end{bmatrix} \tag{5.46}$$

And apply the Newton's correction step:

$$\begin{bmatrix} \widetilde{x}_1^{(k+1)} \\ T_1^{(k+1)} \\ \lambda_1^{(k+1)} \end{bmatrix} = \begin{bmatrix} \widetilde{x}_1^{(k)} \\ T_1^{(k)} \\ \lambda_1^{(k)} \end{bmatrix} - \begin{bmatrix} F_x^{(k)} & F_T^{(k)} & F_\lambda^{(k)} \\ \Pi_q[x_{\Delta t}(\widetilde{x}_0, T_0, \lambda_0)] & 0 & 0 \\ \dot{\widetilde{x}}_0 & \dot{T}_0 & \dot{\lambda}_0 \end{bmatrix}^{-1} \cdots$$
$$\cdots \begin{bmatrix} \Pi_q[F(\widetilde{x}, \lambda, T)] \\ \frac{1}{T}\int_0^T \widetilde{x}_i(t)^T \cdot \dot{\widetilde{x}}_{i-1}(t)dt \\ \frac{1}{T_{i-1}}\int_0^{T_{i-1}} (\widetilde{x}_i^{(k)}(t) - \widetilde{x}_{i-1}(t)) \cdot \dot{x}_{i-1}(t)dt \ldots \\ \ldots + (T_i^{(k)} - T_{i-1}) \cdot \dot{T}_{i-1} + (\lambda_i^{(k)} - \lambda_{i-1}) \cdot \dot{\lambda}_{i-1} - \Delta s \end{bmatrix}. \tag{5.47}$$

When $||H_k(y;\Delta s)|| < tol$ and when $\left|\left| \begin{bmatrix} \widetilde{x}_1^{(k)} \\ T_1^{(k)} \\ \lambda_1^{(k)} \end{bmatrix} - \begin{bmatrix} \widetilde{x}_1^{(k-1)} \\ T_1^{(k-1)} \\ \lambda_1^{(k-1)} \end{bmatrix} \right|\right| < tol$, allocate $(\widetilde{x}_1, T_1, \lambda_1) = (\widetilde{x}_1^{(k_{\text{final}})}, T_1^{(k_{\text{final}})}, \lambda_1^{(k_{\text{final}})})$. (Convergence)

Step 3 end.

Step 4: direction vector update step.

- Normalize the new direction vector.
- Evaluate next tangent direction:

$$\begin{bmatrix} F_x^{(k)} & F_T^{(k)} & F_\lambda^{(k)} \\ \Pi_q[x_{\Delta t}(\widetilde{x}_0, T_0, \lambda_0)] & 0 & 0 \\ \dot{\widetilde{x}}_0 & \dot{T}_0 & \dot{\lambda}_0 \end{bmatrix} \begin{bmatrix} \dot{\widetilde{x}}_1 \\ \dot{T}_1 \\ \dot{\lambda}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.48}$$

- Rescale afterward the new tangent vector to length 1.

Step 4 end.

5.4 Results

In Fig. 5.6 we place ourselves in the context of a Simulink type experiment with noise generated using a Band-Limited White Noise block. This block generates normally distributed random numbers suitable for use in continuous or hybrid systems.[1] To calibrate this noise, we can play with the power spectral density height parameter. We have chosen a value of 3e-04. We can see that our method works perfectly for stationary points. It is important to note that Broyden's method saves an substantial amount of time in the continuation process. In comparison, in Newton's method we are forced to compute a new Jacobian matrix at each iteration and therefore have to extract data at each iteration. In the Broyden's method, this step only needs to be done once per point (in fact, we initialise the Broyden's method with a first iteration of the Newton's method), and then we compute an approximation. On a laptop with Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz processor, a continuation that can take more than an hour with Newton's method will take about fifteen minutes with Broyden's method for the same bifurcation diagram. However, the approximation of the Jacobian matrix can also be a source of error, although it allows us to save a lot of computing time. It may happen that the newly calculated matrix is singular or has "NaN" values. To correct this, at each iteration of Broyden's method we check that the Jacobian is not singular. If it is, the iteration is recalculated using Newton's method.

[1] Available at https://fr.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html

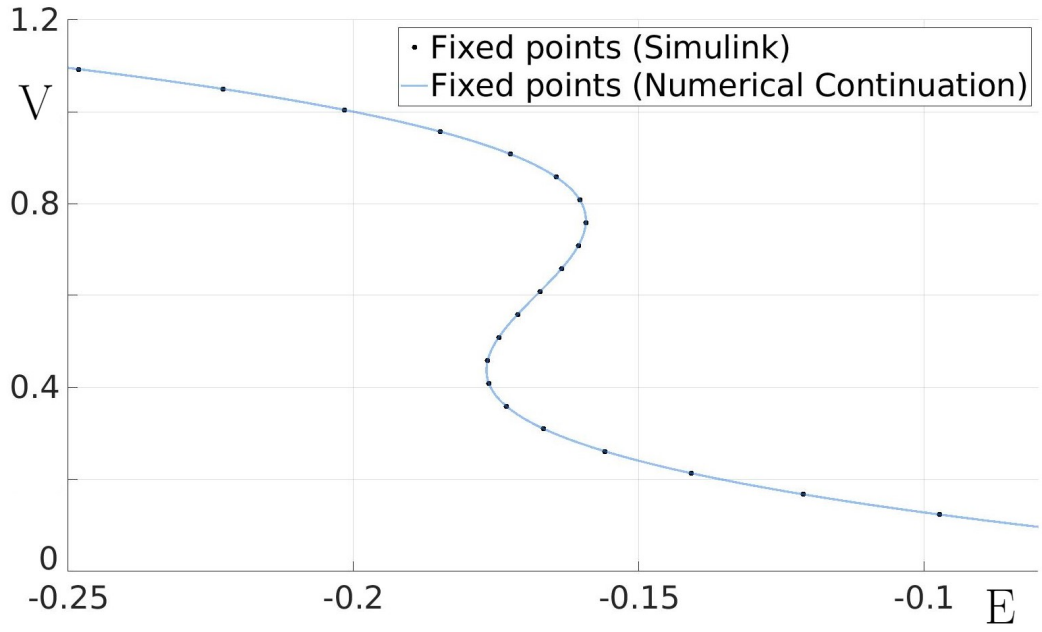

Figure 5.6: Bifurcation diagram obtained from 5.19 with the numerical continuation (blue line) and the CBCE method (black dots). We use the Broyden's method for the correction step and a secant direction vector. The parameters are: $a = 1.8$, $b = 0$, $c = 1$, $d = -1$, $I_{\text{ext}} = 0$, $\varepsilon = 0.6, \widetilde{V}_0 = -3, e_0 = -1.5, ds = 0.05$

We now apply the method to the periodic solution branch. Before considering the FHN model, we consider on a minimal Hopf and saddle-node of limit cycles bifurcation model, namely:

$$\begin{aligned} \dot{x} &= -y + x(\mu + x^2 + y^2 - (x^2 + y^2)^2) + \kappa(x - \widetilde{x}), \\ \dot{y} &= x + y(\mu + x^2 + y^2 - (x^2 + y^2)^2), \end{aligned} \tag{5.49}$$

where μ is the only model parameter. In Fig. 5.7 we see that the CBCE method works very well along the stable and unstable segments of the periodic solution branch. The method also manages to pass the fold of the branch, which assures us that the method is working. It is important to note that the solutions of such a model are circular and therefore simple to compute, since their representation in the Fourier basis is exact with finitely many modes.

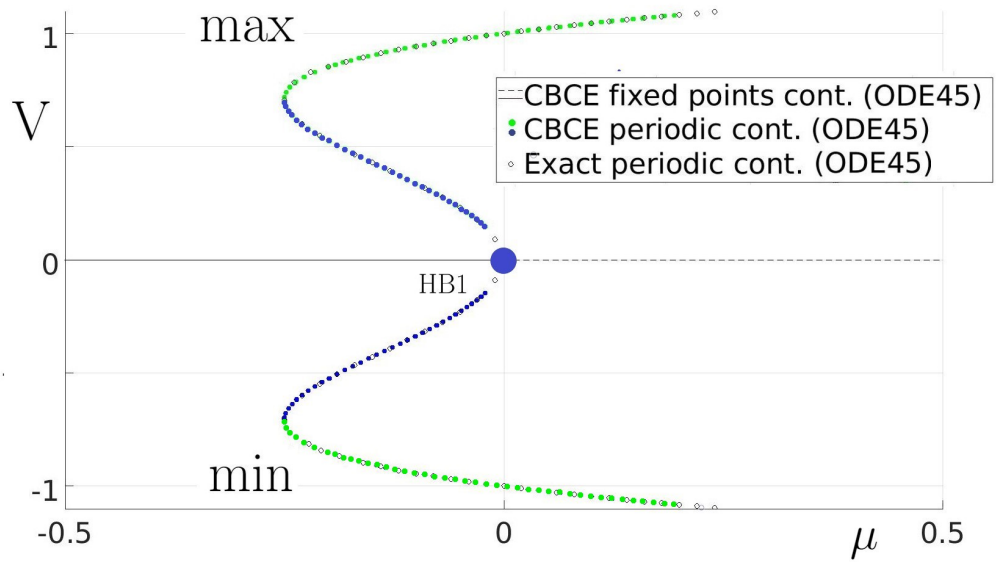

Figure 5.7: Bifurcation diagram obtained for the periodic continuation of 5.49: stable (resp. unstable) families of equilibria (extracted from ODE45) are represented by red (resp. black) dashed lines; stable (resp. unstable) families of limit cycles (extracted from ODE45) are represented with green (resp. blue) dots. The exact periodic continuation for the chosen parameter set is represented with unfilled black dot. HB1 denotes the Hopf bifurcation.

In Fig. 5.8 we will first notice that the shape of the limit cycle branch calculated with ODE45 corresponds qualitatively to that of the numerical continuation obtained for the same model: indeed, we first find two Hopfs points from which branches of unstable periodic cycles emerge, which then each form a fold to give rise to a common branch of stable periodic cycle. However, we will also notice a quantitative type problem, since we observe a diserepancy with the exact diagram. The Hopf points are not found at the same parameter values as the folds. We conjecture that this is due to the fact that we are not using enough Fourier modes. However, increasing the number of Fourier modes makes our implementation of the CBCE algorithm become prohibitise in terms of computational time.

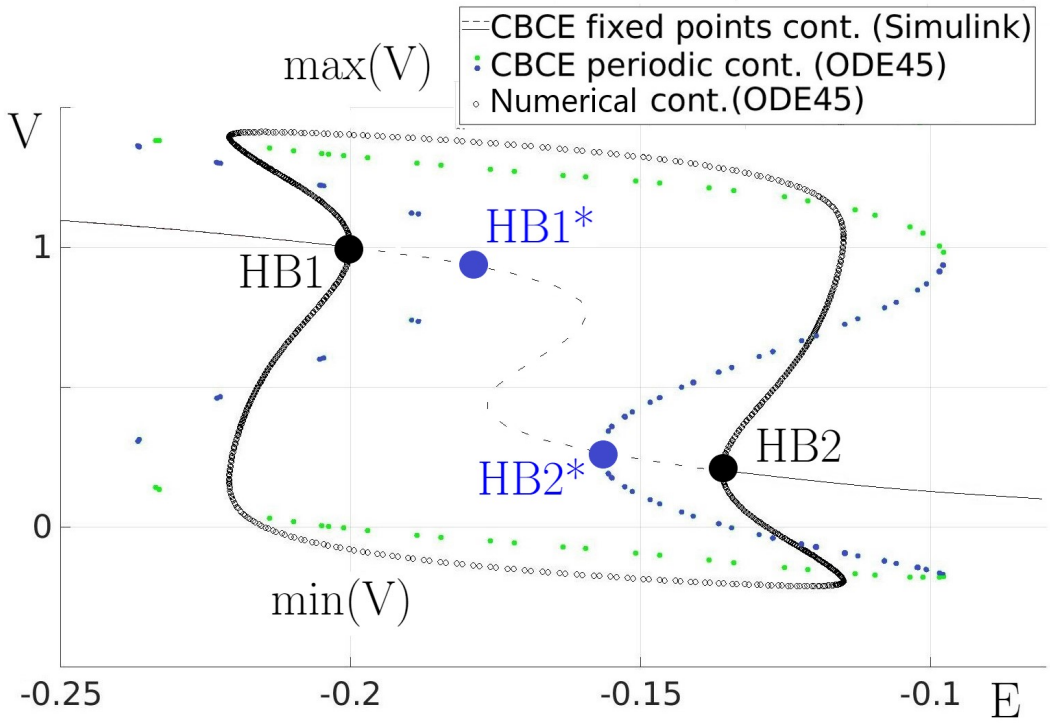

Figure 5.8: Full bifurcation diagram of 5.19. With the CBCE method, we obtained: (1) stable (resp. unstable) families of equilibria (extracted from Simulink) are represented by black solid (resp. dashed) lines; (2) stable (resp. unstable) families of limit cycles (extracted from ODE45) are represented with green (resp. blue) dots. HB1* and HB2* are the two Hopf bifurcations found by CBCE. The periodic numerical continuation is represented with unfilled black dot. HB1 and HB2 are the two Hopf bifurcations obtained with numerical continuation. The parameters are: $a = 1.8$, $b = 0$, $c = 1$, $d = -1$, $I_{\text{ext}} = 0$, $\varepsilon = 0.6, \widetilde{V}_0 = -3, e_0 = -1.5, ds = 0.05$

Following these results, we perform the following convergence test:

$$||F(x+\varepsilon_{\text{conv}}) - F(x) - J(x,h_{\text{conv}})\varepsilon_{\text{conv}}|| = \mathcal{O}(h_{\text{conv}}^2), \tag{5.50}$$

where h_{conv} corresponds to the finite difference step, and $\varepsilon_{\text{conv}}$, a small perturbation parameter which will be varied to see the convergence effect of the finite-difference approximation.

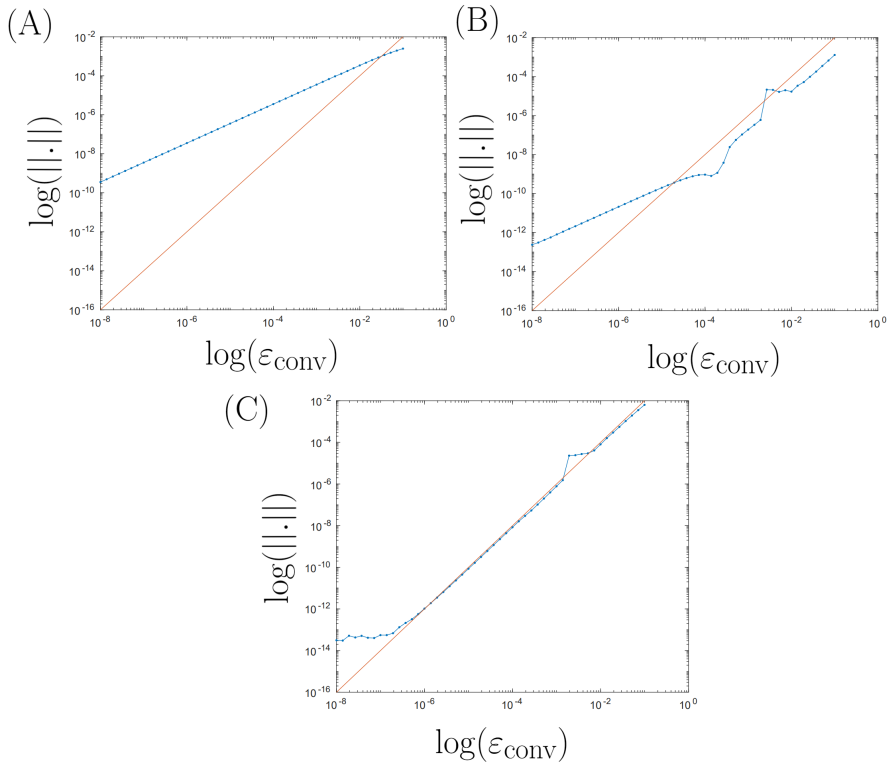

Figure 5.9: Convergence test performed for fixed values of (A) $h_{\text{conv}} = 1e^{-1}$, (B) $h_{\text{conv}} = 1e^{-4}$ and (C) $h_{\text{conv}} = 1e^{-7}$. The orange curve corresponds to a control trace ε,ε^2 where ε, and the blue curve corresponds to the result obtained from the convergence test for the fixed values of h. For better visualization of the data, we propose a logarithmic representation.

What we observe in Fig. 5.9 is that the value of h_{conv} chosen is indeed very important to allow precise derivatives: from panel (A) to (C), the values of h_{conv} are lower and lower, and thus the closer the derivatives of the Jacobian matrix are to the exact values sought. This therefore implies that the convergence is quadratic as expected and that our correction step using Newton's method will therefore be more inclined to converge towards the good continuation values.

In order to understand where this quantitative divergence comes from between the periodic continuation through the CBCE method and the numerical continuation, we propose to calculate the periodic branches with the CBCE method associated with the *harmonic balance method* [153, 154]. Harmonic balance is based on the idea that if a dynamic system is subject to a periodic regime, then its variables can be decomposed into harmonic components, and in our case, these components will be Fourier series terms, i.e. a finite sum of sinusoids and cosinusoids.

Suppose that the solutions of the FHN system are written by a finite sum of n Fourier terms, such that:

$$\begin{aligned} V(t) &= V_0 + V_1 \sin(\omega t) + V_{-1} \cos(-\omega t) + \ldots + V_n \sin(n\omega t) + V_{-n} \cos(-n\omega t) \\ W(t) &= W_0 + W_1 \sin(\omega t) + W_{-1} \cos(-\omega t) + \ldots + W_n \sin(n\omega t) + W_{-n} \cos(-n\omega t) \end{aligned} \tag{5.51}$$

In the case of periodic continuation, we want to find the solution of the problem $\dot{V} - TF_V = 0$ and $\dot{W} - TF_W = 0$, thus, we will calculate the derivatives $\dot{V}$ and $\dot{W}$ from the equations 5.51:

$$\begin{aligned} \dot{V} &= V_0 + V_1 \cos(\omega t) + V_{-1} \sin(\omega t) + \ldots + V_n \cos(n\omega t) + V_{-n} \sin(n\omega t) \\ \dot{W} &= W_0 + W_1 \cos(\omega t) + W_{-1} \sin(-\omega t) + \ldots + W_n \cos(n\omega t) + W_{-n} \sin(n\omega t) \end{aligned} \tag{5.52}$$

Thus, suppose we are studying $q = 1$ Fourier modes, we can then rewrite $\dot{V} - TF_V = 0$ as follow:

$$\begin{aligned}&(V_0 + V_1\cos(\omega t) + V_{-1}\sin(\omega t)) - T\Big(-(V_0 + V_1\sin(\omega t) + V_{-1}\cos(-\omega t))\\&((V_0 + V_1\sin(\omega t) + V_{-1}\cos(-\omega t)) - a)((V_0 + V_1\sin(\omega t) + V_{-1}\cos(-\omega t)) - b)\\&-\frac{(V_0 + V_1\sin(\omega t) + V_{-1}\cos(-\omega t))^3}{3} - (W_0 + W_1\cos(\omega t) + W_{-1}\sin(-\omega t))\Big) = 0\end{aligned} \tag{5.53}$$

And $\dot{W} - TF_W = 0$ can be rewrite:

$$\begin{aligned}&(W_0 + W_1\cos(\omega t) + W_{-1}\sin(\omega t))\\&-T\Big(C(V_0 + V_1\cos(\omega t) + V_{-1}\sin(\omega t))\\&+D(W_0 + W_1\cos(\omega t) + W_{-1}\sin(-\omega t)) + E\Big) = 0\end{aligned} \tag{5.54}$$

With these new equations, it is possible to calculate time series, and to calculate the Fourier coefficients. We thus find the elements necessary to complete the extended problem and the Jacobian matrix of the CBCE method.

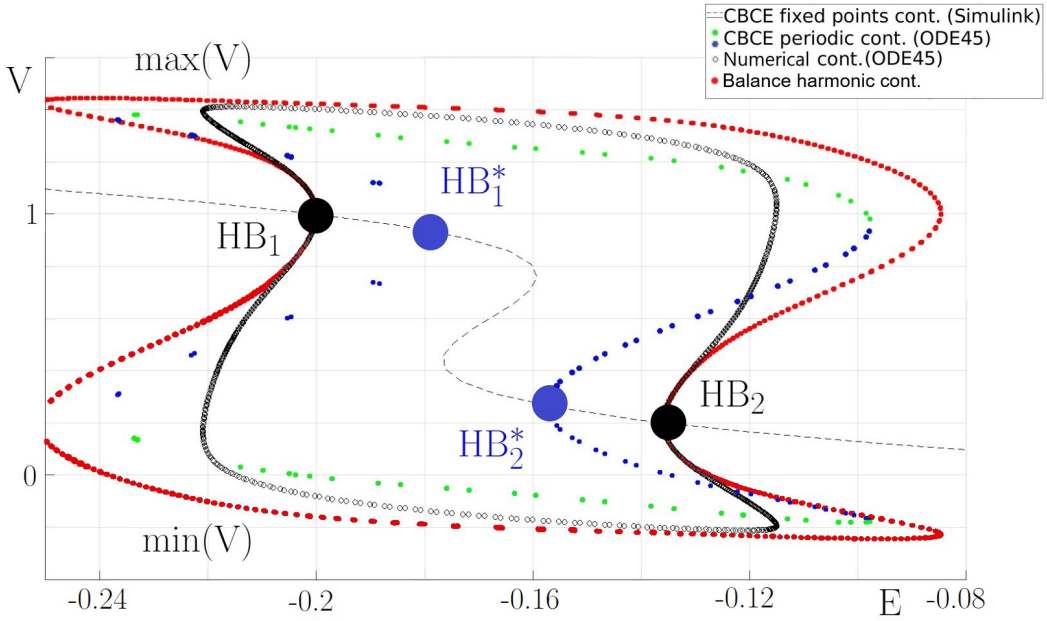

Figure 5.10: Balance harmonic periodic continuation overlaid with the full bifurcation diagram 5.8. With the CBCE method, we obtained: (1) stable (resp. unstable) families of equilibria (extracted from Simulink) are represented by black solid (resp. dashed) lines; (2) stable (resp. unstable) families of limit cycles (extracted from ODE45) are represented with green (resp. blue) dots. HB1* and HB2* are the two Hopf bifurcations found by CBCE. The periodic numerical continuation is represented with unfilled black dot. HB1 and HB2 are the two Hopf bifurcations obtained with numerical continuation. The parameters are: $a = 1.8, b = 0, c = 1, d = -1, I_{\text{ext}} = 0, \varepsilon = 0.6, \widetilde{V}_0 = -3, e_0 = -1.5, ds = 0.05$

In Fig. 5.10, we propose to compare the results coming from the CBCE harmonic balance method, with numerical continuation and the classic CBCE method of periodic branches. First

of all, we notice that the Hopf points of the branch of the CBCE harmonic balance continuation completely match the location of the Hopf points of the numerical continuation. This leads us to think that there is indeed a problem in our calculation of the periodic branch via classical CBCE, and not only in our choice of the number of Fourier coefficients. However, we also notice that the branch of the harmonic balance CBCE continuation has a structure qualitatively similar to that of the classic CBCE method: the structure is extended at the level of the folds than the classic CBCE method, and moves away from the structure resulting from numerical continuation. We can assume here that the problem arises from the number of Fourier coefficients used to calculate the branches. However, we should not discount the fact that our CBCE code may have some flaws, and that it is also very sensitive to these parameters and initial conditions.

5.5 Discussion and conclusion

In this chapter, we were interested in obtaining bifurcation diagrams from "simulated" noisy experiments based on ODE models. The whole point was therefore to have an experiment of an excitable system of which we knew the internal dynamics and therefore the expected solutions, in order to verify our results. These experiments act like black boxes, in that the experimentalist can only interact with them through certain parameters, and only read the output of the experiment.

The CBCE method that we implemented, inspired by the pseudo-arclength continuation method, first allowed us to obtain satisfactory results in the calculation of the branch of stationary points. Regarding the calculation of the periodic branch, the results are more variable. The algorithm seems to have no problem calculating periodic branches so the solutions will have circular shapes (see example of the minimal SNP system). However, in the case of the FHN model, the algorithm seems to have difficulty finding quantitatively the same results as the numerical continuation. After doing some research on the subject, we hypothesized that this problem could come from the choice of the number of Fourier modes used to approximate the solutions of the experiment, but also other subtleties of the code implemented could pose a problem.

In terms of perspective, this project could therefore be completed by research into why we obtain these quantitatively different results from numerical continuation, then when the algorithm is completely established, apply it on neurons *in vitro*.

This work complements the previous chapter, and demonstrates that beyond an approximation, it is possible to obtain the exact bifurcation diagram of an experiment. However, it involves a long processing time, and technical issues when we want to study more complex systems. To conclude these two last chapters, the continuation method of Chapter 4 proposes a method which allows obtaining an approximate bifurcation diagram, but which is easy to use and fast throught a classic Patch-Clamp setup. Conversely, Chapter 5 proposes a more complex method which requires knowledge about continuation, but which attempts to produce the exact bifurcation diagram of the dynamics of the neuron.

Chapter 6

Metastable odotopic representations in mice olfactory bulb

This work was carried out in collaboration with Peter beim Graben [1], Tobias Ackels [2], Andreas Schaefer[2]. I participated in the RSA application, the classification and the result analysis.

Having explored extensively experimental and mathematical approaches to understanding neuronal excitability in previous chapters, we now move towards a more computational perspective. Indeed, the notion of neuronal excitability can be studied through experimental data in order to understand in more detail the neuronal activation patterns, or the regions of interest in the assimilation of a stimulus. This project therefore aims to study neuronal excitability through data analysis methods. This final part of our thesis takes us into the realm of data analysis, where we will closely examine the neuronal responses of the olfactory bulb to various olfactory stimuli. Using the Recurrence Structure Analysis (RSA) method, we will seek to highlight common structures in the excitability of olfactory bulb neurons according to different families of odors and we will build odotopic maps. An odotopic map is a neuronal mapping that represents the relationships between different regions of the olfactory bulb and olfactory inputs, thus reflecting the neuronal representation of olfactory stimuli within the olfactory system. This chapter will allow us to conclude our exploration of neuronal excitability by highlighting the links between experimental data and advanced computational analyses.

6.1 Introduction

Calcium imaging is a technique that allows real-time visualisation and monitoring of neuronal activity by measuring changes in the concentration of calcium ions inside nerve cells: neurons use calcium as a key intracellular messenger to signal electrical activity and synaptic transmission. When a neuron is activated, calcium channels open, allowing calcium to enter the cell. This transient increase in calcium concentration can be detected and quantified using calcium-sensitive fluorescent markers. This method uses fluorescent probes specifically designed to bind to calcium; see Fig. 6.1. The most commonly used fluorescent markers are calcium-sensitive fluorescent proteins (such as GCaMP), which fluoresce when calcium binds to them. These proteins are genetically expressed in target neurons or injected into brain tissue. Real-time monitoring can be performed using special microscopes equipped with high-resolution imaging systems. Calcium-labelled neurons are excited by laser light of a specific wavelength, causing them to emit fluorescence. Detectors then capture this fluorescence, which is converted into digital signals for analysis. Changes in fluorescence over time correspond to changes in calcium concentration, reflecting the electrical activity of the neurons.

[1] Bernstein Center for Computational Neuroscience, Germany
[2] The Francis Crick Institute, United-Kingdom

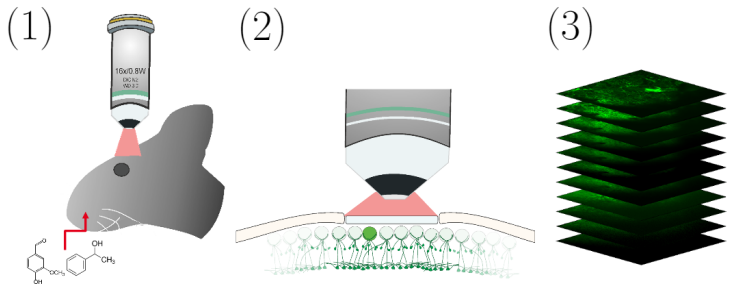

Figure 6.1: Calcium imaging experimental procedure: Once cells are labelled with a sensitive calcium indicator, (1) chemicals are presented to a subject connected to a fluorescence microscope for 30 seconds (2) to record the calcium activity of labelled neurons in the olfactory bulb. (3) This recording is done in layers, making it possible to follow the activity in the olfactory bulb 3D space.

The olfactory bulb is an onion-shaped structure at the base of the brain that processes olfactory information from olfactory receptors in the nasal cavity [155, 156]. It is made up of several layers of neurons, each with a specific role in processing olfactory information. In this chapter we will be interested in *tufted cells*: they play an important role in processing olfactory signals and regulating *mitral cells*, which transmit olfactory signals from the olfactory bulb to other regions of the brain, including the amygdala and the olfactory cortex [157–162]. Indeed, tufted cells can modulate mitral cell activity by acting at different levels: synaptic transmission, intrinsic activity, or synchronisation of neuronal activity. They are also involved in olfactory selection and discrimination by regulating synaptic plasticity between mitral cells. Tufted cells are located in the outer layer of the olfactory bulb, they receive olfactory signals from mitral neurons and are involved in modulating the transmission of olfactory information. Tufted cells have a small cell body and few branched apical dendrites; see Fig. 6.2.

In this chapter, we will use Recurrence Structure Analysis (RSA), introduced in Chapter 1, to unravel the behaviour of tufted neurons exposed to different chemicals: these data come from three mice, whose calcium activity was recorded. Applying the RSA method to these data will involve a number of steps: 1) Developing a pipeline to enable a comparative study of these data, bearing in mind that there is as yet no mapping of these neuronal regions of interest (ROI) [163–165]. We will also want to process the signal to reduce the noise. 2) Parametrising the size of the regions of interest by optimizing the RSA. 3) Using supervised classifiers to certify that the metastable states obtained are significantly indicative of the type of chemical being studied. We will compare several classifiers and try to extract mapping from them.

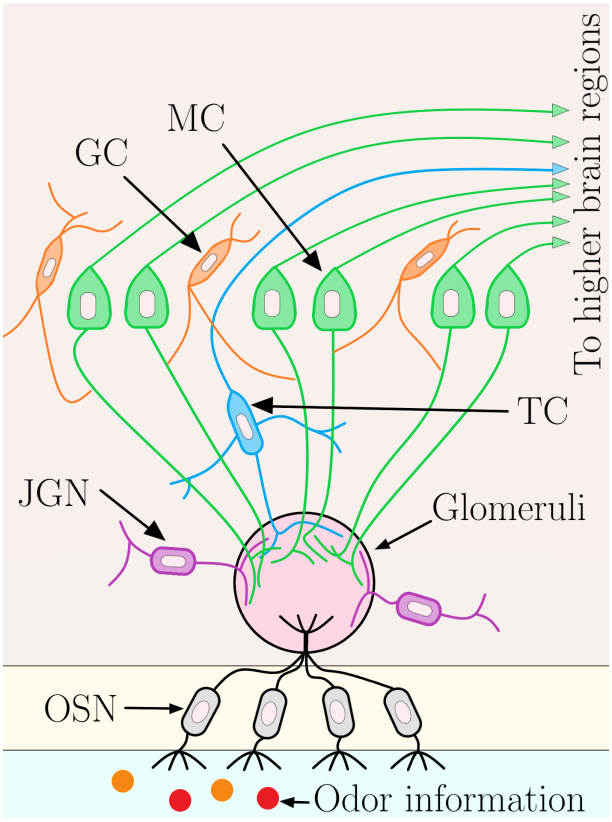

Figure 6.2: Organisation and function of the olfactory bulb: Olfactory sensory neurons (OSNs), located in the nasal cavity, detect the odour molecules present in the air we breathe. Electrical signals are transmitted from each olfactory receptor to an olfactory neuron via a short axon. Olfactory neurons are grouped together in structures called the olfactory epithelium. Axons from olfactory neurons converge to form structures called glomeruli in the olfactory bulb, which is located at the base of the brain. Olfactory information is processed in the olfactory bulb by mitral neurons (MC) and tufted cells (TC). Mitral neurons receive signals from glomerular cells involved in the formation of synapses between the endings of olfactory neurons (olfactory sensory cells) and the dendrites of mitral cells, and transmit them to other regions of the brain, while tufted cells are involved in modulating the transmission of olfactory information. Olfactory signals are then transmitted to regions of the brain involved in processing and analysing sensory information, including the primary olfactory cortex and associated regions of the brain. The granule cells are known as GCs and the juxtaglomerular neurons, JGNs.

6.2 Material and methods

6.2.1 Experimental context and data pre-processing

A series of 48 chemicals (see Tab. 6.1) were presented to three mice (known as subjects ”C525”, ”C531” and ”C537”) for two seconds each, always in the same order, and with enough time to allow them to return to the resting state. A complete recording of an odour lasts 30 seconds. Calcium activity imaging was used to follow tufted cells neuronal activity in the olfactory bulb. Each time series for each neuron is the average of five tests performed on the same mouse at sufficiently spaced times.

The recorded number of neurons for each subject is as follows: C525 has 276 tufted cells recorded, C531 has 195 , C537 has 188. As the number of cells is different in each subject and it is obvious that they cannot be compared due to the notion of individuality, it is difficult to make a clear comparison without pre-processing the data.

Acids (5)	Alcohols (8)	Aldehydes (4)
Nonanoic acid	1-Nonanol	2-Phenylpropion-aldehyde
Hexanoic acid	1-Heptanol	Benzaldehyde
Benzyl acetate	2-methyl-4-butanol	Valeraldehyde
Isobutyric acid	2-methoxy-4-methylphenol	Octanal
Valeric acid	Cyclohexanol	
	Guaiacol	
	Eugenol	
	Cineole	
Aromatics (4)	**Esters (14)**	**Ketones (12)**
1,2-dimethoxybenzene	Ethyl valerate	2-Hydroxyacetophenone
Alpha-terpinene	Ethyl heptanoate	(+)-Fenchone
4-allylanisole	Ethyl butyrate	2-nonanone
R-limonene	Ethyl acetate	2-heptanone
	Cis-3-hexenyl tiglate	Acetophenone
	Propyl acetate	4-phenyl-2-butanone
	Isoamyl acetate	1,4-cineole
	Methyl valerate	5-(+)-carvone
	Methyl salicylate	2,4 dimethyl-acetophenone
	Methyl butyrate	Menthone
	Methyl benzoate	2-hexanone
	Geranyl acetate	4-methyl acetophenone
	Ethyl caproate	
	Ethyl tiglate	

Table 6.1: Chemical signature of the 47 stimuli tested on the olfactory bulb of the three subjects. The 48^{th} is a mineral oil which is an odourless stimulus and so acts as a control.

In order to study and compare subjects, we decided to segment the space of neuron coordinates of all subjects into ROIs using the *K-means* methods [65–67]. The K-means method is an unsupervised machine learning technique used to group data based on similar characteristics. The basic principle of K-means is to divide the data set into K_m homogeneous groups (clusters) by minimising the within-cluster variance and maximising the between-cluster variance. To do this, the algorithm first randomly selects K_m initial cluster centres and then assigns each data point to the closest cluster in terms of Euclidean distance. The data points for each cluster are then averaged to update the cluster centres, and the assignment and updating steps are repeated until the cluster centres converge to their final positions.

The trade-off with this type of method is, of course, knowing what value of K_m to use. If K_m is too small, there will be too few regions to explore, but if K_m is too large, one may end up with regions where a subject does not have any neurons represented (Fig 6.3). Another important question is the maximum number of subgroups that can be created to study individuals among themselve. In fact, the individual with the lowest number of neurons has 188 neurons, while the largest group has 276. We want to avoid this individual being underrepresented compared to other mice by having too many subgroups.

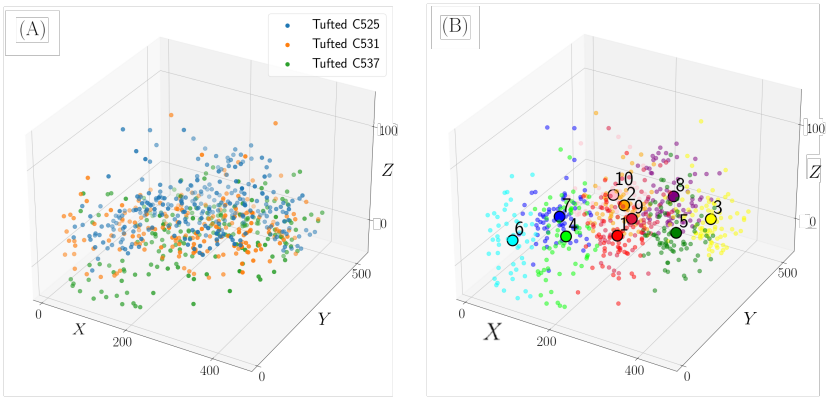

Figure 6.3: (A) Neurons of all three subject represented in (X,Y,Z) space, and then (B) merged into 10 sub-regions using the K-means method. The numbered circles represent the centre of each sub-region.

In order to normalize the data across the experiments conducted, while retaining the phenomenon related to stimulation, we applied a stochastic process called *baseline alignment* [68]. First, we suppose that the stimulation accurs at time $t = 0$ and the signal is described as:

$$x_i(t) = s(t) + \eta_i(t), \tag{6.1}$$

where $s(t)$ is assume to be the invariant signal, and $\eta(t)$ is the noise. We also assume that before stimulation, there is only noise, hence:

$$s(t) = 0 \quad \text{for} \quad t < 0. \tag{6.2}$$

Next, we apply a correction of the baseline by first computing the averages of the prestimulus intervals:

$$\beta_i = \lim_{T\to\infty} \frac{1}{T} \int_{-T}^{0} x_i(t)dt. \tag{6.3}$$

Because of (6.2), we have only noise $\eta_i(t)$ during the pre-stimulation time, so:

$$\beta_i = \lim_{T\to\infty} \frac{1}{T} \int_{-T}^{0} \eta_i(t)dt. \tag{6.4}$$

Then, we subtract that baseline values from the corresponding time series, namely we consider:

$$\zeta_i(t) = x_i(t) - \beta_i = s(t) + \eta_i(t) - \beta_i. \tag{6.5}$$

Using the new signal ζ_i, we first compute its empirical mean (indicated by the sum), and subsequently its stochastic expectation:

$$E(\overline{\zeta_i(t)}) = E\left(s(t) + \frac{1}{N}\sum_{i=1}^{N}[\eta_i(t) - \beta_i]\right) = s(t), \tag{6.6}$$

where N is the number of time series studied for one of the regions, and E, the estimator. This formula is true under the assumption of stochastic ergodicity. Thanks to this method we can thus reduce the noise present in the recordings, while preserving the response to the olfactory stimulus. Fig. 6.4 illustrates the effect of the alignment process.

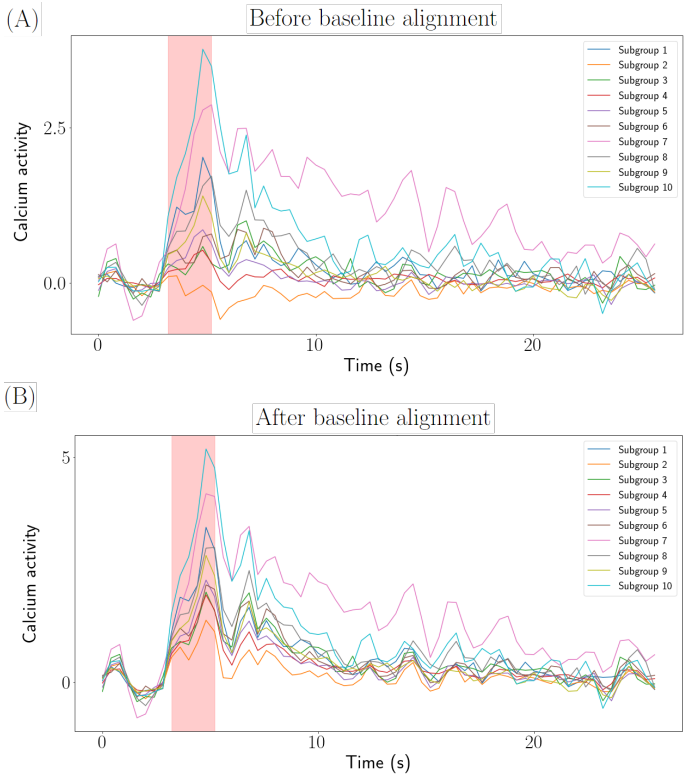

Figure 6.4: Tufted cell time series obtained after stimulation with the Nonanoic acid chemical on the subject C525 (the color coding is the same as in Fig 6.3). (A) Before baseline alignment, (B) after baseline alignment.

6.2.2 Optimal Recurrence Analysis

Consider a trajectory sampled in a discrete way such as $X = \{x_t \in \mathbb{R}^M | 0 \leq t \leq T\}$, where T is the duration and M is the phase-space dimension of the system. In our case, $M = K_m = 10$ and each time series x_i is the tufted cell mean activity in one region of interest. This choice is based on an optimization of the number and significance of metastable states that we obtain in our results.

The recurrence plot (RP), denoted R, is a binary matrix indicating recurrence events time-by-time. Two states x_i, $x_j \in \mathbb{R}^M$ with associated times $i > j$ are said recurrent, when x_j lies in a ball $B_\varepsilon(x_i)$ (for the euclidian distance) of radius $\varepsilon > 0$ centered x_i. Then, R is defined by:

$$R_{ij} = \begin{cases} 1 & \text{if } x_j \in B_\varepsilon(x_i) \\ 0 & \text{else} \end{cases} \tag{6.7}$$

Herein, we use the cosine distance:

$$d_{\cos}(x,y) = 1 - x \cdot y \tag{6.8}$$

for normalized states, $||x|| = ||y|| = 1$.

According to [22, 23], it is possible to rewrite the time series indices by using the RP (6.17) as a rewriting grammar, also called *recurrence grammar*; the procedure is as follow: Suppose two states at time points i and j ($i > j$). If the system is recurrent for the states x_i and x_j ($R_{ij} = 1$), then we will use a grammar rule defined as $i \to j$, which means that we replace, in the sequence s, the larger time index i by the smallest of the recurrent pattern index j. Suppose now that we have one more state x_k such that $i > j > k$. If we have the three states in the same recurrent structure ($R_{ij} = 1, R_{ik} = 1$), then the grammar rules will be $i \to k$ and $j \to k$, and we will replace i, j by the the smallest index k. See Fig. 6.5 for an illustration time sketch. Using this grammar at least twice allows to create and highlight metastable states in the new transformed sequence s'.

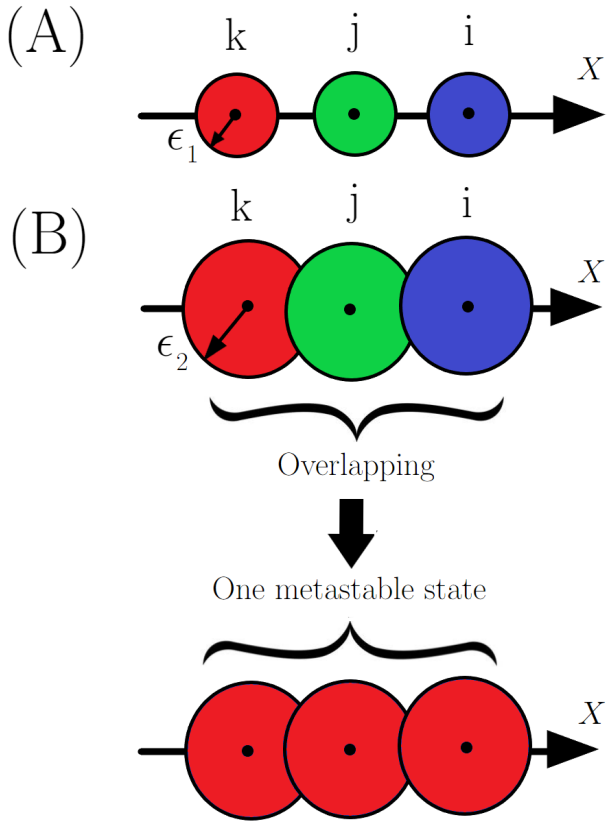

Figure 6.5: Representation of the overlapping phenomenon of spheres of radius ε: (A) ε_1 is not large enough for the spheres of times i, j and k to overlap. Therefore, a metastable state cannot be determined. (B) ε_2 is large enough for the spheres with center times i, j and k to overlap. Thus, times i and j are assigned the same symbol as k. These times are part of the same metastable state.

Therefore, a metastable state, denoted S^k, is defined as a cluster of all states from the trajectory X that have the same index k in the rewritten sequence s':

$$S^k = \{x_i \in \mathbb{M} | s'_i = k\}. \tag{6.9}$$

Thus, it is possible to discretize the phase space and segment X into separated equivalent classes, i.e., the metastable states. However, this segmentation is dependent on the ball size parameter ε. Several methods have already been proposed in order to optimize ε [22, 23], but we will use the Markovian optimization approach [69]. First, we make the assumption that the time series X can be described by a Markov state model, and hence that we can define a transition matrix, $P = (p_{i,j})$, giving the transition from a state S_j into state S_i:

$$p_{i,j}(\varepsilon) = Pr(x_{t+1} \in S_i | x_t \in S_j) \tag{6.10}$$

where x_t is the state at time t and x_{t+1} the state at time $(t+1)$. Another important assumption is that the system spends most of the time in each metastable state. This means that the transitions from one metastable state to another one, passing through a transient regime, are uniformly distributed according to a maximum entropy principle [22, 23]. Combining all these assumptions allows us to write the following utility function:

$$u(\varepsilon) = \frac{1}{n+2}[trP(\varepsilon) + h_r(\varepsilon) + h_c(\varepsilon)] \tag{6.11}$$

where tr$P(\varepsilon)$ is the trace of the transition matrix and

$$\begin{aligned} h_r &= -\frac{1}{\log(n-1)} \sum_{j=1}^{n-1} p'_{0j} \log p'_{0j} \\ h_c &= -\frac{1}{\log(n-1)} \sum_{i=1}^{n-1} p'_{i0} \log p'_{i0} \end{aligned} \tag{6.12}$$

with $h_{r,c}$ denoting respectively the entropy of the transition matrix row and the transition matrix column one, with renormalized transition probabilities

$$\begin{aligned} p'_{0j} &= \frac{p_{0j}}{\sum_{j=1}^{n-1} p_{0j}} \\ p'_{i0} &= \frac{p_{i0}}{\sum_{i=1}^{n-1} p_{i0}} \end{aligned} \tag{6.13}$$

An optimal partition is then obtained through

$$\varepsilon^* = \arg\max_{\varepsilon} u(\varepsilon), \tag{6.14}$$

involving a maximally metastable Markov state model. The "complexity" of the transition model, including the transient regime and $n-1$ metastable states, is defined by the number of segments denoted $n(\varepsilon^*)$ [69].

Consider now that we are studying the ensemble of N trajectories $E = \{N | 1 \leq m \leq N\}$, which in our case refers to time series from $N = K_m = 10$ regions of interest. Because the data are normalized to a unit hypersphere, we can compare the metastable states present in the different region of interest. To do so, we need first to collect all metastable states $S_k^{(m)}$ from all individuals, and then calculate their pairwise Hausdorff distances:

$$D_{ij} = \max\{\max\{\delta(y, S_j) | y \in S_i\}, \max\{\delta(y, S_i) | y \in S_j\}\} \tag{6.15}$$

according to [70]. The distance between the point x and the compact set $A \subset X$ is computed as follow:

$$\delta(x,A) = \min\{d(x,y)|y \in A\} \tag{6.16}$$

Note that the Hausdorff distance of two overlapping compact sets vanishes. Again, we use the cosine distance (6.8) for the normalized data here.

Then, imposing a threshold to the distance matrix D with respect to a parameter $\theta > 0$ gives a new matrix denoted Q which contains the results from the Hausdorff clustering of the metastable states based on a new recurrence grammar [70]. Namely:

$$Q_{ij} = \begin{cases} 1 & \text{if } D_{ij} < \theta \\ 0 & \text{else} \end{cases} \tag{6.17}$$

It is important to choose carefully θ in order to cover with a minimal set of metastable states the entire ensemble E. Thanks to this methodology, we can therefore study and compare the recurrence plots that come from different subjects or different experimental contexts. In the present context, it will be pertinent to compare the results of a stimulus on the three subjects, as well, the results of all stimuli on one given subject.

6.2.3 Classifiers

The results obtained from RSA, allow us to check whether metastable states for each chemical are sufficiently significant to be used as a discriminator in a classifier. Since we have access to the label of each of the studied chemicals (acid, alcohol, aldehyde, aromatic, ester, ketone, control), we focus only on supervised methods.

The *K-Nearest Neighbour*[3] (KNN) algorithm is a machine learning method used to classify data [166–168]. It searches for $K_{\text{neighbours}}$ instances in the feature space that are closest to the new instance (for which we want to predict the class) and determines the majority class among these $K_{\text{neighbours}}$ neighbours. The distance between instances can be measured using different metrics, and in our case we will use the cosine dissimilarity distance. The choice of the value of $K_{\text{neighbours}}$ is important for the performance of the algorithm. Indeed, if $K_{\text{neighbours}}$ is too small, the model may be too sensitive to noise in the training data and not generalise well to the test data. Similarly, if $K_{\text{neighbours}}$ is too large, the model may be too insensitive to variations in the data and not capture the relationships between features and targets well enough. We will also use the Support Vector Machine (SVM) method [169–171] and the Stochastic Gradient Descent (SGD) method [172] for comparison with the KNN method.

To improve the classifier results, we will use over-sampling methods. The bootstrap method [173–177] is the most basic process to generate multiple samples of the original population by performing sampling with replacement, i.e. each observation in a bootstrap sample may be selected multiple times, and some observations may not be selected at all. Repeating this process of bootstrapping results in a collection of samples that conserves the distribution of the original population.

The Synthetic Minority Over-sampling Technique (SMOTE) is also a widely used over-sampling method designed to address class imbalance in supervised learning datasets [178–180]. It aims to generate synthetic examples of the minority class to create a balanced dataset [181]. The algorithm achieves this by creating new examples by interpolating feature values from nearest neighbours of minority class examples. The basic steps of the SMOTE algorithm are to select an example from the minority class, randomly select k nearest neighbours from the minority class, and generate a synthetic example by combining the feature values of the selected example and its neighbours.

[3] Available at https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

This is done by calculating the differences between the feature values of the selected example and each neighbour, multiplying these differences by a random number between 0 and 1, and adding the weighted product to the selected example, resulting in a new synthetic example. These steps are repeated until the desired number of synthetic examples has been generated. SMOTE effectively increases the size of the minority class by creating synthetic examples along the decision boundaries between classes. This allows the characteristics of the minority class to be better captured and reduces the risk of overfitting the existing data. However, care must be taken when using SMOTE as it may introduce noise due to the generation of synthetic examples.

Finally, to determine which classifiers to study, we used the *accuracy*, *precision* and *recall* as test scores [167]. We will also examine the confusion matrix of classifiers to visualise the correct and incorrect predictions for each class. This allows us to highlight the classes that are most difficult to classify.

6.3 Results

In this study we will use $K_m = 10$ ROIs. We want to be able to compare the response of the three subjects to each chemical (Fig. 6.6). So we applied the RSA method to the collected data of each mouse for the different chemicals, and we applied the Hausdorff method to unify the colours of the metastable states when studying the same odour class. This means that if we study the case of eugenol, each yellow metastable state is the same state in all three bands. In this figure, we first notice a general structure through all the chemicals due to our chosen K_m parameter value: at $t = 0$, before the stimulation, a first metastable state is present and it corresponds to the rest state of the system. Then we have a transient which gives way to a reaction specific to the chemical presented. Subsequently we have another transient and finally a return to the resting state. This general structure therefore tells us that we have calibrated the RSA method well and that we can now study the different structures specific to the different chemicals. It is important to note that this demonstration of similar structure between the three subjects is not always possible because of experimental bias or because of the individuality of the subjects.

In terms of the reaction itself, we note that in the post-stimulation it was possible to highlight similar structures of metastable states in the three subjects at the same time. In the case of eugenol, methyl salicylate and 5-(+)-carvone, the structure specific to each of these chemicals is extremely similar across the three subjects. In the case of 4-allylanisole, ethyl valerate and (+)-fenchone, the structures are also similar, but the Hausorff algorithm may not recognise one subject's metastable state as being the same as that of another subject. However, we can still emphasise that these metastable states generally occur at the same time. These differences can also be explained by the variability of the recordings or the subjects studied.

Nevertheless, these results imply that our spatial segmentation into $K_m = 10$ regions of interest for the three subjects via their coordinates does indeed make it possible to highlight that certain regions have effects on the dynamics of the system. We now want to know which regions of interest are the most active or important in representing metastable states, and thus be able to compare across the three subjects whether this pattern is common or not.

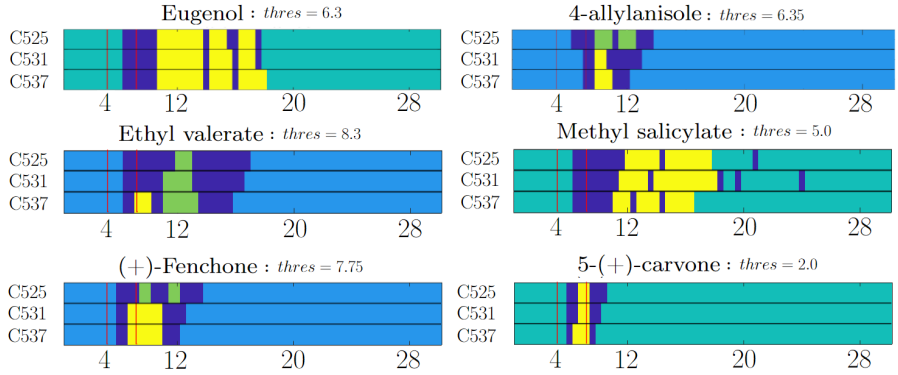

Figure 6.6: Example of the results of the RSA on all the tufted cells of the three subjects, for 10 subgroups. The Hausdorff threshold is adapted for each individual odour in order to highlight common states: it is thus possible to compare the colours of the recurrence states only in the same odour type. The x axis represents time in seconds. The two vertical lines are the time limits of the stimulation, which is applied for two seconds. (Parameters: $ds = 20$, $pr = 15$)

.

To this aim, we then extracted the centroids of each of the metastable states and plotted them, for the example of methyl salicylate, first on a simple graph (Fig 6.7 (A)) and then on a 2-dimensional heatmap representing only the x and y axes of the spatial coordinates of the tufted neurons (Fig 6.7 (B)). (Fig 6.7 (B)) Thus, in panel (A) we can see that the two metastable states have exactly the same characteristics in terms of centroids. Indeed, for the first metastable state, which corresponds to the rest state, the centroids seem to be very low compared to the second, but it seems that the regions of interest 7 and 8 are higher than the others. For the second metastable state, we see that the same regions of interest are highlighted as the most active, again for the three subjects. The heatmaps in panel (B) show the same thing and even allow us to study the response in a spatial way. Once again, we see that the three subjects have the same regions of the olfactory bulb that are activated during the presentation of methyl salicylate. Moreover, comparing the metastable states of one subject gives us the intuition that the pattern is indeed similar, but with different intensities. Thus, the neural activation vectors appear to be almost collinear.

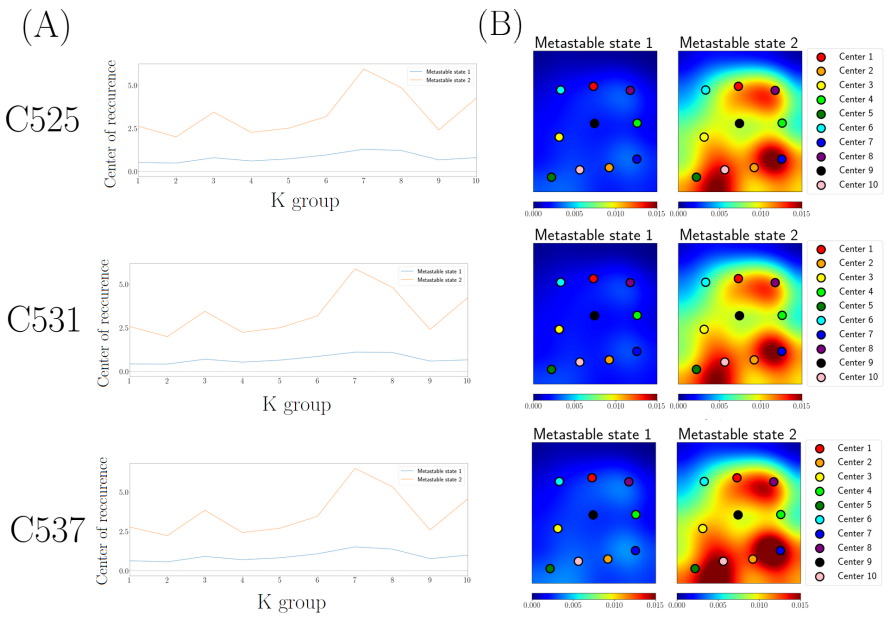

Figure 6.7: Study of the centroids extracted from the RSA results for the methyl salicylate recurrence domains: (A) we first plot the centroids of the metastable states in a simple graph to have the exact values, and then (B) we plot the same information but in a 2D heat map where the two axes are the x and y spatial coordinates of the tufted cells. To design and smooth the heat maps, we used a Gaussian filter with a standard deviation value of $\sigma = 60$ (parameters: $ds = 20$, $pr = 15$).

It is important to note that the use of the K-means method is random, although the number of regions of interest to be studied is fixed, which can lead to variations. However, although there are variations (not the same ROI labels or not the same neurons in each region), the metastable state structures as well as the centroids are still very similar across the 3 subjects. We now want to extract the centroids for all chemicals and subjects in order to classify them.

In this section, we want to classify the 48 chemicals into the correct odour class among the 7 that are present (acid, alcohol, aldehyde, aromatic, ester, ketone, control). We have concatenated the centroids presented in the last section in a matrix which has a first dimension of 10 (clusters) and a second dimension which is the product between the number of subjects, the number of chemicals and the corresponding number of metastable states highlighted by RSA. In the present case we have a second dimension of 308 (points). Now we want to determine which method is the best to classify our data.

First of all, we have the labels associated with each point of our data set, which makes the use of a supervised classifier obvious. We therefore tried different classifiers of this type (Stochastic Gradient Descent, Support Vector Machine and K-nearest neighbours): we used a training set (75% of the population) and a test set (25% of the population) to apply these methods and found that the KNN method gave the best results with an average accuracy of 44.155% (Tab. 6.2, K-nearest neighbour parameters: $K_{\text{neighbours}} = 5$).

The new goal is to improve this accuracy by preprocessing the data. To do this, we propose two variants of widely used techniques in machine learning: the bootstrap and the SMOTE methods. (see section 6.2.3) The bootstrap method is used to increase the size of our dataset to 500 points (which represents 162 % of the initial population), while maintaining part of the proportionality of the dataset. We then applied the various KNNs again, and our average accuracy improved to 71.2% with the K-nearest neighbour method (K-nearest neighbour parameters: $K_{\text{neighbours}} = 5$). Noteworthy, in the data set studied, not all odours are sparsely represented. Therefore, in order to increase the average accuracy and to balance the categories in the tufted case, we used the SMOTE

Method	Average accuracy (%)
KNN + SMOTE method	**74.14 %**
KNN + Bootstrap method	**71.2 %**
KNN	**44.155 %**
SVM (linear kernel) + Bootstrap method	56.396 %
SVM (linear kernel)	41.403 %
SVM (polynomial kernel)	32.596 %
SGD	31.016 %

Table 6.2: Average accuracy of the different supervised classifiers used to study the concatenation of the recurrence domain centres of all the chemicals studied, obtained for the case of tufted cells. The average is calculated over 100 trials. The training set for each tested classifier represents 75% of the population. KNN: K-Nearest Neighbours, SVM: Support Vector Machine, SGD: Stochastic Gradient Descent.

method to obtain 85 samples from each category by creating synthetic samples (the oversampled dataset represents 192.5% of the original dataset), and we obtained an accuracy of 74.14% with the KNN method (K-nearest neighbour parameters: $K_{\text{neighbours}} = 5$). This accuracy is really good in the context of a data set coming from biological experiments.

From now on, we will choose to study the KNN classification method, with or without preprocessing methods. The choice of the value of K_m is made in order to find a compromise between having enough neighbours and having good score values for our tests.

To see if our results are valid, we then decided to compare the results obtained for the same KNN method, with their accuracy and their precision. (Fig. 6.8) These results support us in the idea that the classifiers associated to the two preprocessing methods (SMOTE and Bootstrap) are not content to classify randomly, but give a good performance in their way of learning and classifying from the results obtained with the RSA algorithm. (The parameters of the applied KNN method: $K_{\text{neighbours}} = 5$).

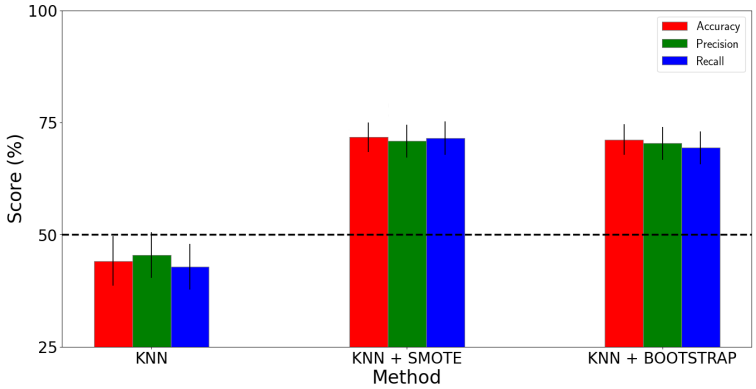

Figure 6.8: Average accuracy, precision and recall obtained from the K-nearest neighbour tests with the different processed datasets (raw, SMOTE and bootstrapped). The training set for each tested classifier represents 75% of the population. The standard deviation is also shown and calculated over 30 trials.

We also suggest studying the normalised confusion matrices (Fig. 6.9) of the three data sets shown in Fig. 6.8. These matrices reveal for which odour classes we have misclassifications. If we do not pre-process the data, we can see that we are indeed not able to correctly classify the samples in the good categories. If we take into account the bootstrap method, it gives better results due to the presence of the maximum of predictions on the diagonal. However, the control case is

never well classified. Regarding the SMOTE case, the results are even better than the bootstrap one, in the sense that there is a less important spread of bad predictions. It also seems that the control case is better classified in these two cases.

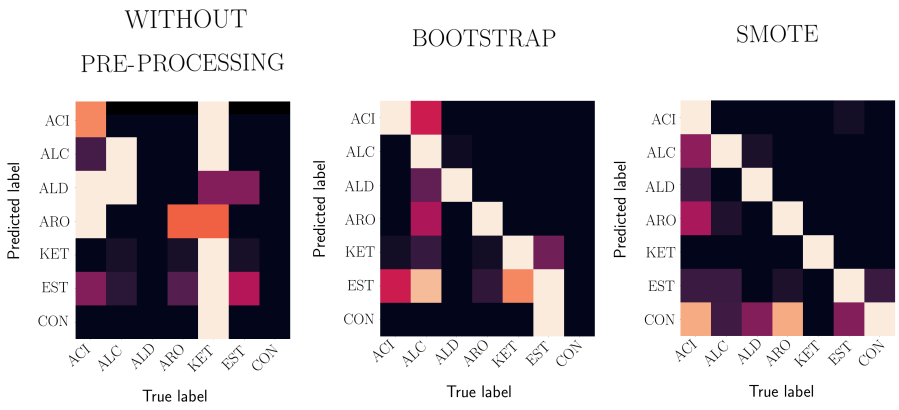

Figure 6.9: Confusion matrices obtained for the data without processing, the data processed with bootstrap, and with SMOTE method. The matrices are normalized with the max/min method for each row independently. The labels are defined as follow: acids (ACI), alcohol (ALC), aldehyde (ALD), aromatic (ARO), ketone (KET), ester (EST) and control (CON).

Finally, we propose to recover the good predictions of the classifier associated with SMOTE and to average these points in 10 dimensions for each category, then to develop the heat maps of the centroids of each of our populations (Fig 6.10). This allows us to see which are the most important points for each type of odour class in the olfactory bulb by eliminating bad predictions or outliers. It should also be noted that the SMOTE + KNN algorithm does not sort the data correctly, so there are only a few points to be considered in this example. Fig. 6.10 (A) shows that some of the odour classes seem to have different centroid scales (for example the ketones compared to the alcohols). The subgroup number 2 seems to be really active for each type of chemical presented, and that for the three cases.

Next, we will normalise the neuron type cases to their side between 0 and 1. This will allow us to see which group is more active than the others, and to see the patterns for each of the odour classes. In Fig. 6.10 (B) we can see that the same groups of neurons are mainly used when presenting an odour class (subgroups 5, 7, 9 and 10). However, there is a difference in response intensity. Thus, for the case of ketones and aromas, there is a great activity of the groups of neurons 7, 9 and 10, whereas for the control case there is almost no activity.

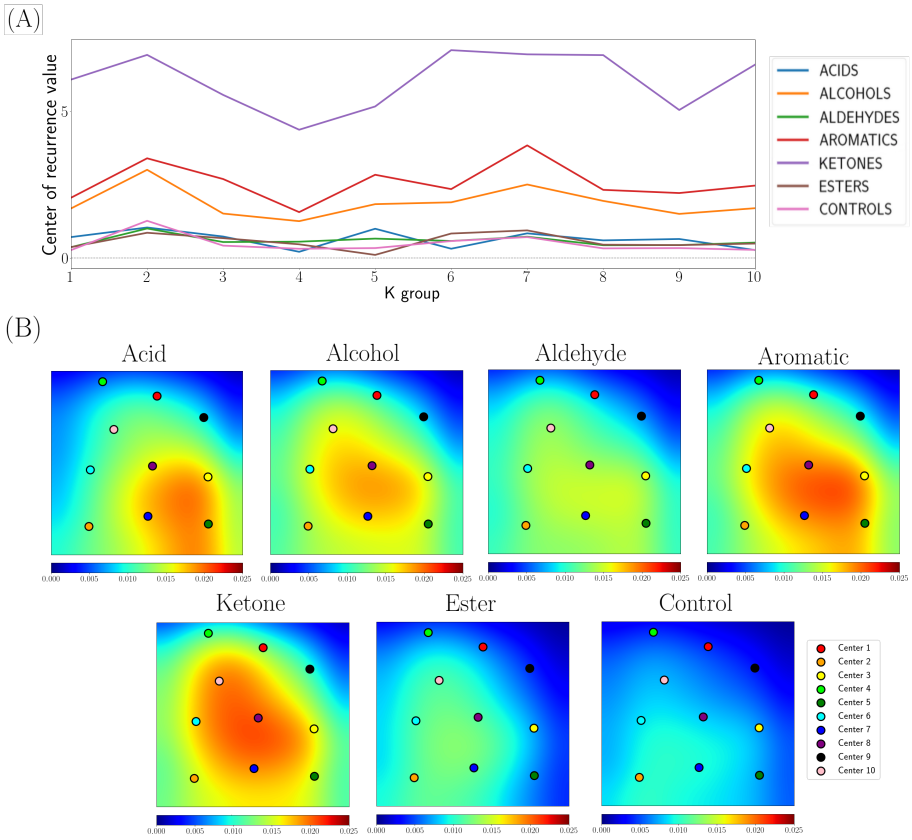

Figure 6.10: (A) Representation of the average of the non-normalized centroids for each odor class of the predictions made correctly by the KNN classifier, with SMOTE type pre-processing and (B) the representation is in a 2D space and give the importance of the clusters for all the odor classes, where the coordinate points of the neurons are only represented on the X and Y axes. The data-sets are normalized separetly between 0 and 1 to understand what are the most active groups spatially. Each colored point is exactly the same population in every panel. To design this heat-map, we used a Gaussian filter with a standard deviation value of $\sigma = 60$.

6.4 Discussion

In this study, we have tried to develop a pipeline that allows us to compare the activity from the olfactory bulbs of three different subjects, for which we have no known mapping. We used different methods that require certain settings. For example, the K-means technique requires a choice of the number of regions of interest to be created from the original set.

It is also interesting to take a critical look at the data used. It is always important to remember that data quality can also be affected by factors such as biological variability, experimental error, measurement bias or sample processing artefacts. We found certain cases where we could not detect a metastable state with the RSA method, but this is not a problem because we had several subjects to generate each chemical metastable state with RSA. In addition, the odour classes are not presented sparingly, which makes it difficult to compare results between categories, but also to train neural networks in our supervised classifiers. Indeed, it is important to consider that if the amount of data is not evenly distributed between the different categories, it can bias the results of the analysis. In our case, if we have a large amount of data for a particular category, it is possible that the prediction model will be more efficient for that category, while being less efficient for the other categories. Taking all this into account, the first thing we can do is to increase the size of the training data by studying the centroids (For a given chemical, we will have several metastable

states ranging from 1 to 4). Enlarging a data set using the bootstrap method can be a useful approach to improve the performance of a classification algorithm, but it should be noted that this method also has some limitations. It is important to understand that the bootstrap method creates a new oversized dataset by taking samples from the original dataset, and it also preserves the proportionality of the data in each category. This means that some observations may be sampled several times, while others may never be sampled. However, it is statistically unlikely that this bias will cause the results to be substantially skewed.

It is also important to note that the bootstrap method cannot create new information from existing data. In other words, if the original data set does not contain enough information to solve the classification problem, the bootstrap method will not solve the problem. However, after carefully increasing the size of the new dataset so as not to over-represent the already large number of samples, we found that the classifiers were able to learn very well from this dataset, even though they couldn't with the original dataset.

The use of the SMOTE algorithm is a good complementary study tool, as it allows us to introduce synthetic examples in the minority classes in order to rebalance the classes. The use of this method thus mitigates the overfitting caused by the random oversampling that bootstrapping can introduce, but also prevents the loss of information due to random selection. Of course, this method also has disadvantages, which we have tried to reduce as much as possible. Firstly, SMOTE can have difficulties with datasets of too high a dimension, but our dataset has only 308 samples in 10 dimensions, distributed across the different odour classes, which makes SMOTE easier to use. The second disadvantage is due to the algorithm itself, in that SMOTE does not take neighbouring samples into account when generating new examples, which can increase the overlap between classes as we increase the size of our dataset. That's why we tried to balance the classes carefully. Thus, thanks to these resampling methods, while minimising the risk of overfitting, we were able to obtain satisfactory results from the olfactory bulb dataset studied.

Finally, we also used an algorithm to automate the construction and optimization of the classifier learning pipeline [182–184]. This algorithm, called the Tree-based Pipeline Optimisation Tool (TPOT)[4], uses scalable algorithms to automatically explore a pipeline search space, selecting and configuring the most appropriate data pre-processing steps and machine learning models. With a cross-validation generated over 5 iterations, in the case of tufted cells with a SMOTE-type preprocessing, better results can be obtained (average accuracy reaching up to 80%), but these involve the entanglement of several classification methods, each with its own set of parameters. Therefore, in this study, we concluded that we should focus on simpler supervised classification techniques that give equally good results.

6.5 Conclusion

In this study, we analysed with olfactory neuron data from three mice, focusing on the study of tufted neurons. Our main aim was to see if it was possible to identify whether certain groups of neurons were more active than others in recognising an odour class across individuals. This involved developing a pipeline to map the olfactory bulbs of the three subjects to allow comparison. To do this, we used a method that combines several data processing techniques: first, we used the K-means method to group the data from each mouse into clusters. Then we applied a procedure called baseline alignment, which allowed us to reduce the noise in our recordings. We then used the RSA method to extract recurrent patterns from our data. We extracted information about the importance of neuron subgroups in each of the metastables for each of the chemicals and subjects. We used this information as training data for our supervising classifiers. Using the K-nearest neighbour method to classify our data, the results were more than convincing for studying

[4] available at http://epistasislab.github.io/tpot/

data from experiments. The classification scores showed that our method is effective in comparing neuron activity between mice.

Through this pipeline, we were able to construct maps that allow us to compare the intensity of the activity of the different subgroups of the olfactory bulb across the different subjects. These representations allow us to hypothesize that the areas of the olfactory bulb requisitioned for the recognition of chemicals are the same for each subject. Furthermore, we were also able to show that these structures will have the same recurrence patterns and therefore the same way of processing information for a given chemical.

In conclusion, our method, which combines the K-means, RSA and KNN methods, is effective to firstly allow the comparative study of subjects when no mapping is known, to highlight which neuron group is most active following olfactory stimulation and to determine a discriminative feature for learning a classifier for chemical recognition. Overall, this study provides new insights into the analysis of olfactory neuron data and demonstrates the effectiveness of the proposed method for future studies in this area. This chapter is therefore part of the thesis as an approach to the study of neuronal excitability through the prism of data analysis, and at the scale of the neural network.

Chapter 7

General Discussion

In conclusion, this thesis proposes an in-depth study of neuronal excitability through different approaches, ranging from dynamical systems to laboratory experiments with computational explorations. Each chapter contains novel results on neuronal excitability, as well as validation of mathematical models as representations of experiments.

In Chapter 2, we undertook a theoretical study, revealing how an integrator type neuron can have its activity modulated so as to display subthreshold oscillations, like a resonator neuron, by applying a forced current through a dynamical-clamp type protocol. This work therefore made it possible to understand that although a neuron is categorized in one way, it can still support a type of behavior that is not characteristic of it. This information could help us to understand with more perspective the transmission of information *in vitro*, but also the plasticity that a neuron can have when external currents are transmitted to it. Furthermore, this chapter demonstrated that a neuron model can serve as a basis for the development of protocols that are suitable for experimental application. It would be so interesting to test experimentally the theorical predictions we have obtained. In addition, we plan to study the propagation of the electrical signal from such a resonator-like integrator neuron through a network of standard integrators.

In Chapter 3, we studied the "flipper" behavior of GCs of the dentate gyrus. This work focused first on the study of the results obtained by our experimental colleagues who induced these cells to display flipping via a specific sodium current, then on the development of a Hodgkin-Huxley type model based on this specific sodium current to demonstrate it also exhibits this kind of behavior. Future work on this project could be to understand the biological role of this unexpected behavior in GCs that transitions between immature and mature states.

In Chapters 4 and 5, we focused on obtaining bifurcation diagrams from noisy experiments via numerical continuation methods. Based upon closed-loop feedback control, with an embedded rootfinder algorithm (e.g. Newton's method), the CBCE method allows to overcome the fact that we do not a priori know the governing equations associated with the experiment we are studying. In chapter 4, we studied experimental data obtained by subjecting a neuron to two different patch-clamp electrophysiology protocols: a voltage clamp with slow ramp on the hold voltage (VC), and a current-clamp with slow ramp on the hold current (CC). The result of the VC protocol provided a (current, voltage) curve resembling a steady-state bifurcation diagram. What is more, once superimposed onto the result of the CC protocol, the two datasets interacted in a similar manner as a full systems solution overlaid onto a fast-subsystem steady-state bifurcation diagram. That is, a slow-fast dissection effect. In this chapter, we explained why this simplified method works, and we pose hypotheses on the question of the noise obtained around the unstable parts of the critical manifold. Thus, this method, which is akin to a simplified revision of CBCE (without the Newton part) allows us to obtain a viable approximation of an experimental steady-state bifurcation diagram, and also it can validate the underlying neuron model as a good representation of the real neuron. In the near future, we would like to apply this method to more types of neurons in order to show the robustness of the method.

In Chapter 5, we focused on applying a complete version of the CBCE method on experiments simulated via Simulink. This complete version is closer to accurate results when it comes

to obtaining curves of stationary points (regardless of the model simulated as an experiment in Simulink). Regarding branches of periodic solutions, the method can calculate them easily when the oscillations are "simple enough", i.e, in normal form type models. However, when the oscillations are more nonlinear as in the FitzHugh-Nagumo model, the method seems to have more problems in approximating the correct branch. With the harmonic balance method, we were able to pose new hypotheses as to the origin of this discrepancy: in fact, these results show us that the problem can come from the number of Fourier coefficients that we use to approximate the solutions, or else, certain subtleties coming from the algorithm itself. In the near future, we would to further explore the reason behind this discrepancy, and then apply it on a real neuron.

Chapter 6 presented a data-analytic study on the olfactory bulb of mice. Namely we analysed calcium imaging data coming from 3 mice subjected to different chemicals. We developed a pipeline to study the data: first we segmented the real coordinate space of the neurons with the k-means method, then we produced the average of the neurons of each segment for each individual. Then, we pre-processed the data with the baseline alignment method, and then applied the RSA method. These first results showed us that there are indeed patterns of recurrence of these olfactory recognitions. Next, having segmented the data into subgroups based on the coordinates of neurons in the olfactory bulb allows us to compare the centroids of these metastable states. These results are summarized in odotopic maps which demonstrate that the same areas of the olfactory bulb are activated for the same chemical. In addition, we also demonstrate that for the different types of chemicals the same areas are used for recognition but at different intensity levels. Finally, we propose the learning of a supervised classifier of these centroids to show that they are significantly sufficient to discriminate them. These results could be supplemented with more complete data, either by having more individuals, or with more trials for each chemical.

During this PhD, we also carried out an analysis of a Morris-Lecar analog neuromorphic circuit (Appendix 8) in order to try to apply our algorithms for continuation or integrator/resonator transition, in the context of computer-controlled closed-loop experiments. In open-loop setup, the experiment worked well, however, with the closed-loop setup we encountered problems that could not be fixed. Namely, delays between the application of the given current to the experiment, and reading the output. The circuit analysis results are presented in the appendix to this thesis.

This thesis therefore shows that the study of neuronal excitability through a combination of mathematical, computational and experimental approaches is crucial for understanding the functioning of the nervous system. Mathematics provides the essential framework for modelling the complex behaviour of neurons, allowing us to generate abstract representations and quantitative predictions that guide our experiments. Computation enhances this ability by providing us with sophisticated tools to simulate and analyse models, and to explore datasets. Experiments provide valuable and tangible data that ground our understanding in biological reality. Combining these approaches is still an ongoing research topic, with novel results up for grab, allowing us to contribute to the understanding of neural activity. The CBCE method is an example of contribution as a way to bridge the gap between models and experiments, reveal unstable neural states and their role.

Chapter 8

Appendix: Morris-Lecar analog circuit dynamical study

8.1 Introduction

In this Appendix, we focus on the study of a neuromorphic circuit based on the Morris-Lecar model. A neuromorphic circuit is a type of electronic circuit designed to reproduce the operation of neural networks in the human brain: this includes the generation of action potentials, synaptic transmission, and other processes electrical and chemical linked to neuronal activity. The interest in studying a single-neuron neuromorphic circuit in this thesis was to have a controlled experiment in which we know the equations which lead the behavior of the model. What we hoped for was to be able to apply our methodologies developed during the thesis on this circuit in order to calibrate them, but also to work on the notion of real-time experience, which was a new challenge for these methods. Indeed, the use of methods based on control theory, such as those seen in chapters 2, 4 and 5, are dependent on a cell voltage feedback term. The presence of a bad interaction in terms of time for this term obviously leads to false results. If this chapter is only present in the appendix, it is because we encountered this difficulty with our equipment and we were unable to resolve this problem. However, as this work is in line with the study of neuronal excitability from an experimental point of view, we decided to present the study of the circuit in this thesis.

In this chapter, we will therefore first present the Morris-Lecar model used, the electronic circuit and its configuration, then the results obtained with it.

8.2 Material and methods

8.2.1 The Morris-Lecar neuromorphic circuit

There are several advantages to studying an electronic circuit of which we know the exact components compared to studying the dynamics of a neuron in vitro: in the case of an electronic circuit, we can completely control all the components and parameters, which allows high precision in measurements and experiments. On the other hand, neurons are complex biological systems and it is difficult to control their dynamics. Furthermore, electronic circuits are simple mathematical models of physical systems, which facilitates the analysis and understanding of their behavior. On the other hand, neurons are complex biological systems and modeling their dynamics is often very difficult. Moreover, electronic circuits are often less expensive to manufacture and test than in vitro studies of neurons, because there is no need to maintain cell cultures or use expensive laboratory equipment. And finally, electronic circuits can be easily modified to explore different hypotheses and scenarios, while in vitro studies on neurons can be limited by technical constraints and experimental protocols.

Of course, electronic circuits are abstract models of physical systems, which do not always accurately reflect the complexity of the real world. Therefore, conclusions drawn from the study of electronic circuits may not apply to other more complex systems. Moreover, they are often

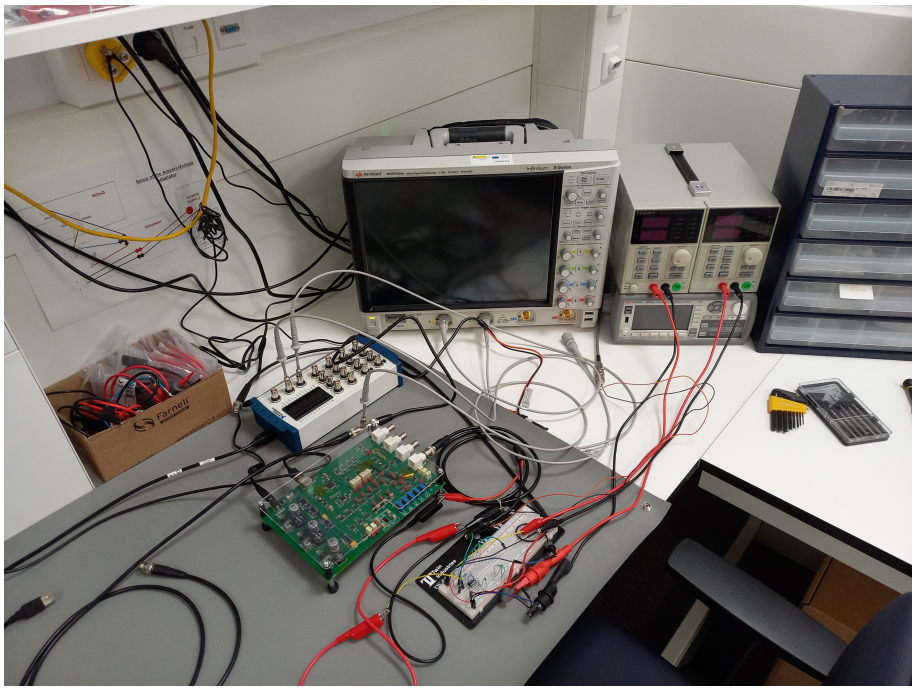

Figure 8.1: General setup.

simplified models of physical systems, which do not take into account all interactions and potential influences. Therefore, the conclusions drawn from the study of electronic circuits may be limited by this oversimplification. However, this does not pose a problem for this thesis project since what we want is to study the dynamics of an experiment whose mathematical nature we know.

The neuromorphic circuit is based on a modified version of the Morris-Lecar model [185]:

$$\begin{aligned} C\dot{V} &= -g^*_{Ca} M_\infty(V_m)(V_m - V_{Ca}) - g^*_K N(V_m - V_K) - g_L N(V_m - V_L) + I \\ \dot{N} &= \tau^{-1}(-N + G(V_m)) \end{aligned} \tag{8.1}$$

where V_m is the membrane voltage, N is the activation variable of the slow potassium channels, and I is an external tonic current delivered to the neuron. g^*_{Ca} and g^*_K are the maximal conductances of the calcium and potassium channels, respectively, and g_L is a constant leak conductance. V_{Ca}, V_K and V_L are reversal potential of relevant ion species. Parameter C is the membrane capacitance and τ, the time constant associated with N.

The nonlinear functions $M_\infty(V_m)$ and $G(V_m)$ are defined as follows:

$$\begin{aligned} M_\infty(V) &= 0.5(1 + \tanh\left(\frac{V_m - V_1}{V_2}\right), \\ G(V) &= 0.5(1 + \tanh\left(\frac{V_m - V_3}{V_4}\right), \end{aligned} \tag{8.2}$$

where V_1, V_2, V_3 and V_4 will be considered as adjustable and tuneable parameters for steady state and time constant. In fact, the circuit has two outputs (V_m, the membrane voltage, N, the second variable of the Morris-Lecar model, i.e. the recover variable) and one input (I_{ext}, the external current). It is possible to modify the conductances and the ion equilibrium potentials from the circuit physically, while the parameters V_1, V_2, V_3 and V_4 will be modified directly from a suitable software. The fact that all the parameters cannot be modified via the appropriate software is a disadvantage, however, it does not prevent from producing the results we are looking for.

x	g_x
I	+502.5Ω
L	+500.5Ω
K	+126.7Ω
Ca	+250.1Ω
S	+286.3Ω

Table 8.1: Table of conductances used during the experiments for each of the ions.

x	V_x
Ca	+1.955V
K	-0.729Ω
L	-0.516Ω
S	-0.4433Ω

Table 8.2: Table of equilibrium potential used during the experiments for each of the ions.

8.2.2 Additional informations on the setup

For our experiments, we used a data acquisition device produced by National Instruments. The model is NI 781003-01 - USB-6212 BNC Bus Powered M Series, and it allows a communication between the computer and the circuit. It can send a potential to the circuit and read back the potential coming from the Morris-Lecar output. However, it can send only a Voltage signal going from $-5V$ until $5V$. [1] This device is supposed to be able to support real-time interactions between the PC and the experience.

We also used an homemade electronic board which allowed us to convert the potential coming from the NI-DaQ into a current signal for the Morris-Lecar analog circuit. It has a resistance of $1k\Omega$, and a conversion table can be find bellow (Table 8.3). It is possible to change the resistance on the electronic board. If we chose to use a resistance of $1k\Omega$, this is because when designing the circuit, the company used this set-up to convert the incoming potential. This electronic board is power supplied by a double station device.

8.2.3 Studying the circuit dynamic

In this subsection, we provide a summary of the set-up needed to capture the dynamic a Morris-Lecar analog circuit.

In Fig. 8.2, the protocol main steps about how to use the circuit are explained: (1) with Python, we send the voltage value we want the NIDaQ to generate, (2) and at the same time, we calibrate the Morris-Lecar parameters via the laboratory computer. (3) the generated voltage is sent to the electronic board to be converted in current, (4) meanwhile it is powered by the power supply. (5) This current is sent to the Morris-Lecar circuit and (6) to be sure about the current we are sending to the circuit, we check with the electrometer the exact current value produced. (7) Through the oscilloscope, we can read the Morris-Lecar dynamics, and with the NIDaQ connection, we can also read the membrane voltage input on the computer. This configuration will be kept throughout the experiments presented below.

[1] https://www.ni.com/es-es/support/documentation/dimensional-drawings/model.usb-6212.html

INPUT (Voltage)	OUTPUT (Ampere)
-5V	-1.358mA
-4V	-1.086mA
-3V	-0.814mA
-2V	-0.521mA
-1V	-0.269mA
0 V	+0.002mA
+1V	+0.269mA
+2V	+0.521mA
+3V	+0.814mA
+4V	+1.086mA
+5V	+1.358mA

Table 8.3: Conversion table from the electronic board input (Voltage) into the current output (Ampere).

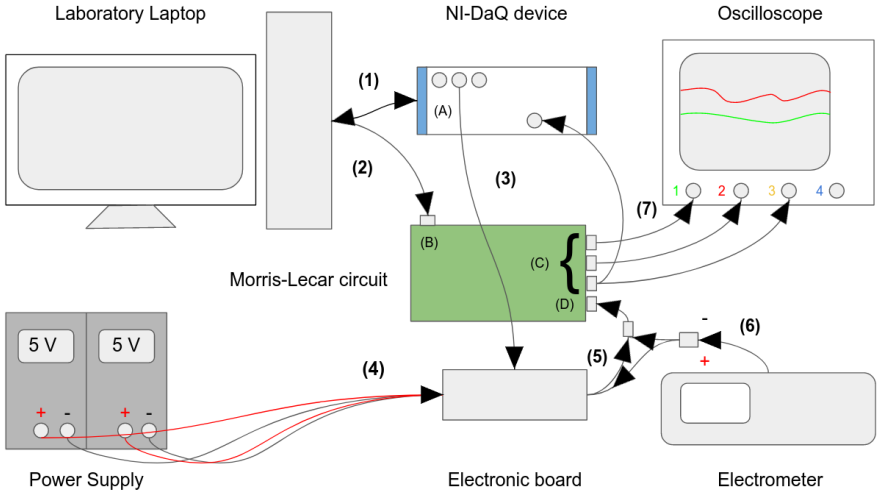

Figure 8.2: Minimum set-up to study the Morris-Lecar dynamics with the circuit. (A) is the data acquisition card which generates the potential via a Python command. (B) is the plug which allows the computer to calibrate the circuit, (C) are all the outputs of the circuit, and (D) the current input.

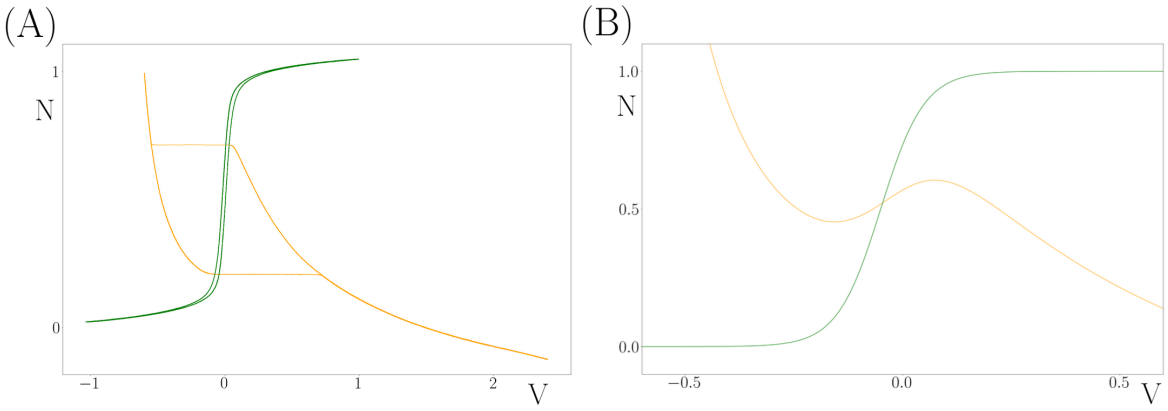

Figure 8.3: A: experimental nullclines. Parameters values are: $I = -1$ mA, $V_1 = 0.0$ V, $V_2 = 4.477$ V, $V_3 = 0.0$ V, $V_4 = 4.441$ V, $\tau = 3.78$ ms, $C = 1.025$ µF, $g^*_{Ca} = 2.508$ mS, $V_{Ca} = 1.955$ V, $g^*_K = 1.253$ mS, $V_K = -0.729$ V, $g_L = 1.0$ mS, $V_L = -0.516$ V, B: model nullclines. Parameter values are: $I = 0.0$ mA, $V_1 = 0.0$ V, $V_2 = 0.15$ V, $V_3 = -0.05$ V, $V_4 = 0.02$ V, $\tau = 2$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V.

8.3 Results

8.3.1 Dynamical behavior

During the first experiments on the circuit, were aimed at highlighting the behaviors and the most common mathematical objects that can be obtained with the Morris-Lecar model.

Thus, we obtained the experimental nullclines shown on Fig. 8.3 A. These experimental nullclines are obtained from a ramp dependent on one of the two variables, which means that this dynamic is off, while measuring the equilibrium value of the other variable. This method has the drawback of not allowing the calculation of unstable branches of the nullcline. They can be compared with the ones obtained through the model; see Fig. 8.3 B. It is hence possible to obtain a N-nullcline very similar from the model one, but the experimental V-nullcline is not exhibiting this unstable part. Indeed, the solution jumps at the nullcline folds due to the ramp method.

It was also possible to highlight robust behaviors of the 8-shaped solution (which effectively displays two canard segments) featuring stable focus points near the V-nullcline folds and a saddle point. In this studied case, we then have a canard solution which displays this behavior at the two folds. This type of scenario is also possible to obtain in the Morris Lecar model, however it is difficult to obtain. Indeed, when the sigmoid nullcline passes close to the folds of the cubic nullcline, it is possible to have these stable focus points, but because the model is not symmetrical, the model requires precise calibration.

Canard solutions have the property of existing over a very small range of parameter values, which justifies the term canard explosion [53, 80]. Thus the noise of the parameters of the electronic circuit allows us to observe the appearance of canard solutions for well-chosen base values. Indeed, an alternation between small canard solutions and large canard solutions is observable in Fig. 8.5 A. Moreover, it is also possible to see that these data show a slight trend of 8-shaped solution.

We have applied a rotation from the coordinates of the data, in order to mimic the observed 8-shaped experimental traces.

To obtain the Fig. 8.5 (C) and (D), we applied a rotation matrix on the differential equation system (eq. 8.1):

$$\begin{bmatrix} \dot{V}_r \\ \dot{N}_r \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{N} \end{bmatrix}, \tag{8.3}$$

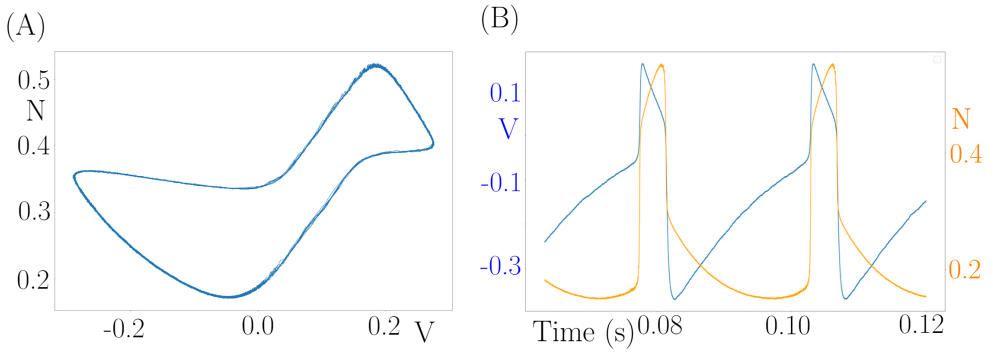

Figure 8.4: Example of experimental double-canard behavior. (A): experimental phase plane projection of the measurement. (B) time series for both measured output of the circuit, namely V and N. Parameters values are: $I = -0.8319$ mA, $V_1 = 0.0$ V, $V_2 = 4.44$ V, $V_3 = 0.0$ V, $V_4 = 4.44$ V, $\tau = 6.27$ ms, $C = 1.025$ µF, $g^*_{Ca} = 2.508$ mS, $V_{Ca} = 1.7$ V, $g^*_K = 1.253$ mS, $V_K = -0.729$ V, $g_L = 1.0$ mS, $V_L = -0.516$ V.

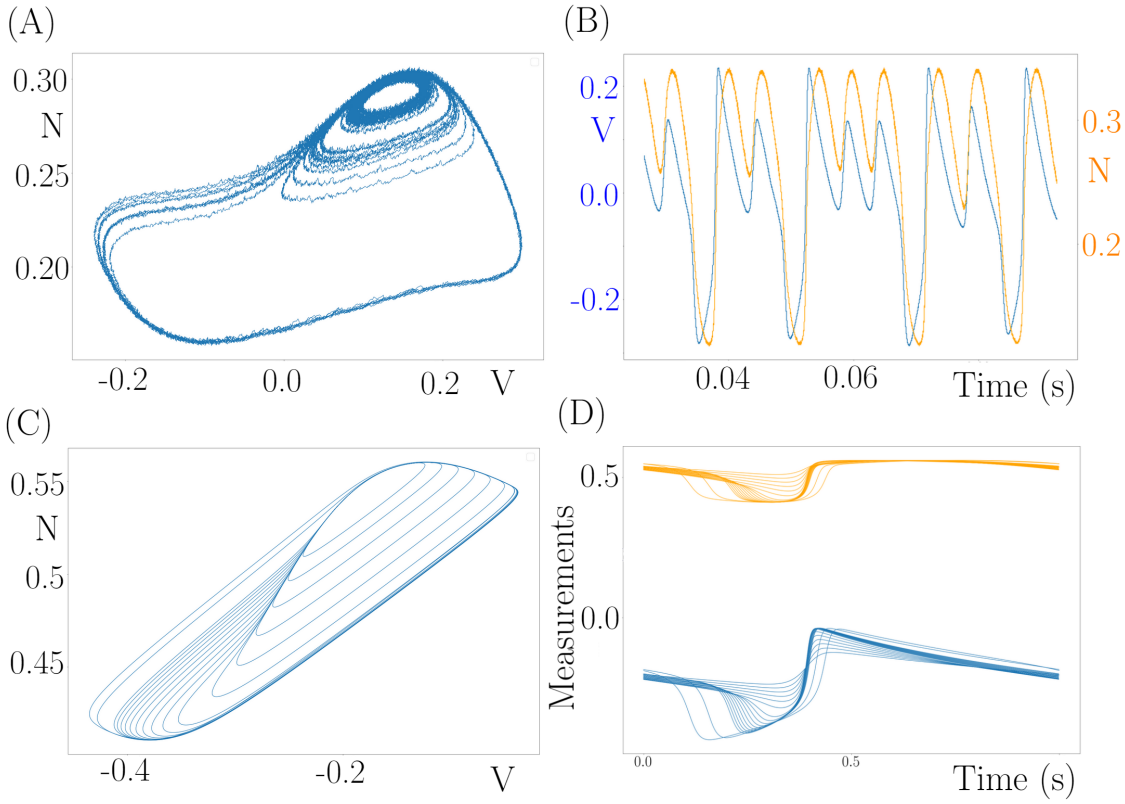

Figure 8.5: (A) Experimental apparition of canard behavior phenomena on a phase plane, and (B) the corresponding time series, (C) Apparition of canard behavior phenomena on a Morris-Lecar model phase plane (Rotated with a value of $\theta = 0.4$ radian). (D) Corresponding time series for each cycle. Parameters values are: $I = 0.5$ mA, $V_1 = 0.0$ V, $V_2 = 1.0$ V, $V_3 = 0.0$ V, $V_4 = 1.0$ V, $\tau = 0.7$ ms, $C = 1.025$ µF, $g^*_{Ca} = 2.508$ mS, $V_{Ca} = 1.992$ V, $g^*_K = 1.253$ mS, $V_K = -0.729$ V, $g_L = 1.0$ mS, $V_L = -0.516$ V.

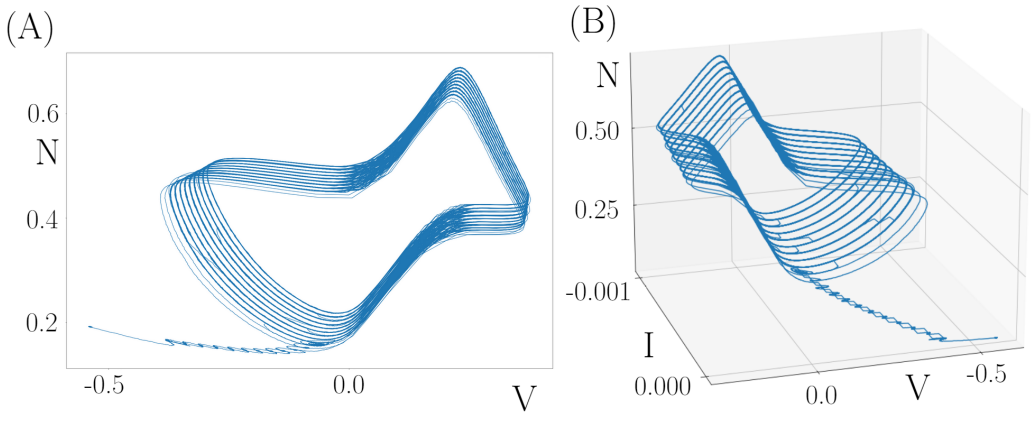

Figure 8.6: (A) phase plan of a circuit experiment involving a rampe for the applied current. (B) 3D phase plan of the same conditions. Parameters values are: $I = 0.0$ mA to -1.0 mA, $V_1 = 0.0$ V, $V_2 = 4.44$ V, $V_3 = 0.0$ V, $V_4 = 4.44$ V, $\tau = 3.78$ ms, $C = 1.025$ µF, $g^*_{Ca} = 2.508$ mS, $V_{Ca} = 1.992$ V, $g^*_K = 1.253$ mS, $V_K = -0.729$ V, $g_L = 1.0$ mS, $V_L = -0.516$ V.

where θ corresponds to the angle of the rotation. This matrix defined in eq. (8.3) rotates points through an angle θ with respect to the positive x-axis about the origin of a two-dimensional Cartesian coordinate system. The final rotated system is as follow:

$$\begin{aligned} \dot{V}_r &= \cos(\theta)\dot{V} - \sin(\theta)\dot{N} \\ \dot{N}_r &= \sin(\theta)\dot{V} + \cos(\theta)\dot{N} \end{aligned} \tag{8.4}$$

We used an increasing ramp for the current applied on the circuit (Fig. 8.6 (A) and (B)), which allows us to highlight the presence and the effect of the Hopf bifurcation point present in the system (Fig. 8.7).

In order to show that this behavior is possible even in a model, we also used a FitzHugh-Nagumo model, i.e. a simplified model of the Hodgkin-Huxley model, which can have a symmetrical phase portrait; this makes it easier to get the behavior. In Fig. 8.7, for both cases, it is possible to observe 8-shaped canard solution, due to the presence of Hopf bifurcations leading to canard explosion near each fold of the critical manifold.

This kind of behavior can only appear for a small range of parameter, so to study it we computed the bifurcation diagram of these models.

In Fig. 8.8, a bifurcation diagram of the Morris-Lecar model is provided according to the parameter V_3, with also a zoom on the parameter range which is giving the particular behavior which can observe in Fig. 8.7 B. Two families of unstable cycles (blue line) are observable, coming from each Hopf bifurcation, HB1 and HB2, and both terminating at two homoclinic bifurcations, HM1 and HM2, involving a saddle point lovated on the middle branch of the critical manifold. The family of large and stable cycles (green line) terminates on both sides at homoclinic bifurcation with the associated homoclinic connections being large, each of them involving a saddle equilibrium (for two nearby values of V_3) located on the middle branch of the critical manifold.

In Fig. 8.8 C, the bifurcation diagram of the FitzHugh Nagumo model, according to the parameter b, is showing also where the 8-shaped solution can appear in this model: from the Hopf points HB3 and HB4, unstable periodic cycle branches born, then undergo a canard explosion, and stop at the point HM5 and HM6. At HM7, a new branch of limit cycle born and it is still unstable, before to become stable: it is during this unstable branch that appear this 8-shaped solution.

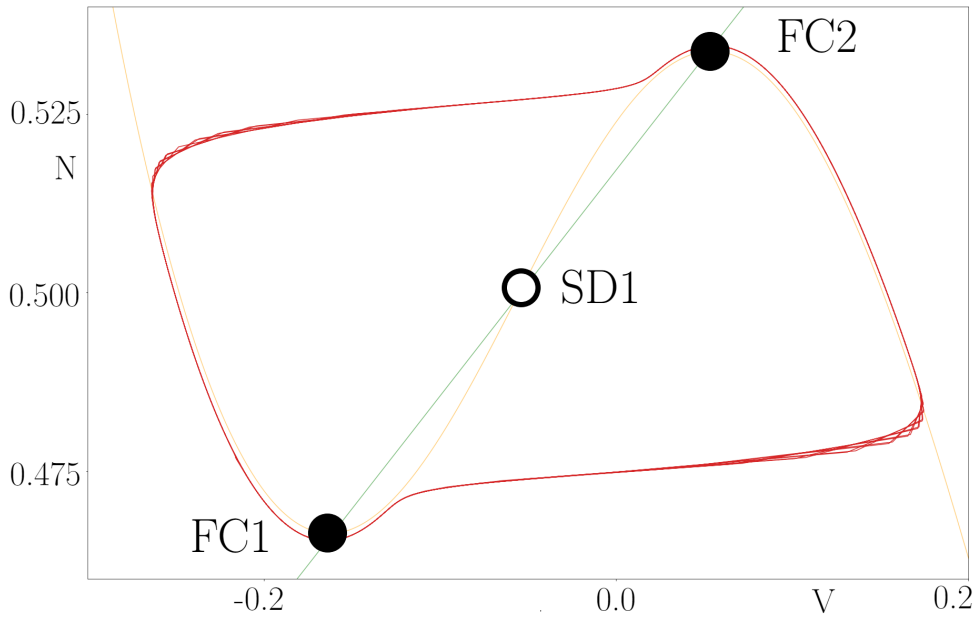

Figure 8.7: Morris-Lecar phase plan with small canard solutions. Orange curve: V-nullcline, green curve: N-nullcline. Parameters values are: $I = -0.23395$ mA, $V_1 = 0.0$ V, $V_2 = 0.2$ V, $V_3 = -0.0542782$ V, $V_4 = 1.581236360973563$ V, $\tau = 10$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V.

8.3.2 Sensitivity analysis

Sensitivity analysis is a mathematical technique used to assess the impact of parameter variations on the results of a mathematical model, particularly in the context of systems of differential equations. In a system of differential equations, the variables often depend on unknown parameters. Sensitivity analysis explores how variations in these parameters influence system results. More precisely, the sensitivity analysis makes it possible to quantify the relative importance of each parameter for the variation of the solution of the system of differential equations.

For this analysis, we use the package Uncertainpy[2], which is a Python library for sensitivity analysis and optimization of stochastic models, based on Bayesian inference methods. This library is used to calculate sensitivity indices for stochastic models from numerical simulations. It also provides tools for optimizing model parameters and for performing statistical analyzes on simulation results. Uncertainpy supports several types of stochastic models, such as Markov processes, diffusion processes, and jump processes.

In Fig. 8.9, we propose a sensitivity analysis of the system according to two parameters V_3 and V_4, which are involved in the N-equation, and to the parameter set which is giving us the particular double canard behavior. The methodology used implements polynomial chaos expansions using point collocation method. This polynomial chaos expansion method have support for the rosenblatt transformation to handle dependent input parameters. It is feature based, i.e., if applicable, it recognizes and calculates the uncertainty in features of the model, as well as the model itself. In our study, the two parameters follow a uniform distribution of ± 1 or ± 0.1 around their basis value. It is possible to observe that the 8-shaped cycles are not robust, even with a small variation.

8.3.3 Attempt to apply our methods

In this section, we wish to illustrate two problems encountered when applying our methods on the Morris-Lecar circuit. First of all, we want to try to apply our slow and controlled forcing on

[2] Available at https://uncertainpy.readthedocs.io/en/latest/

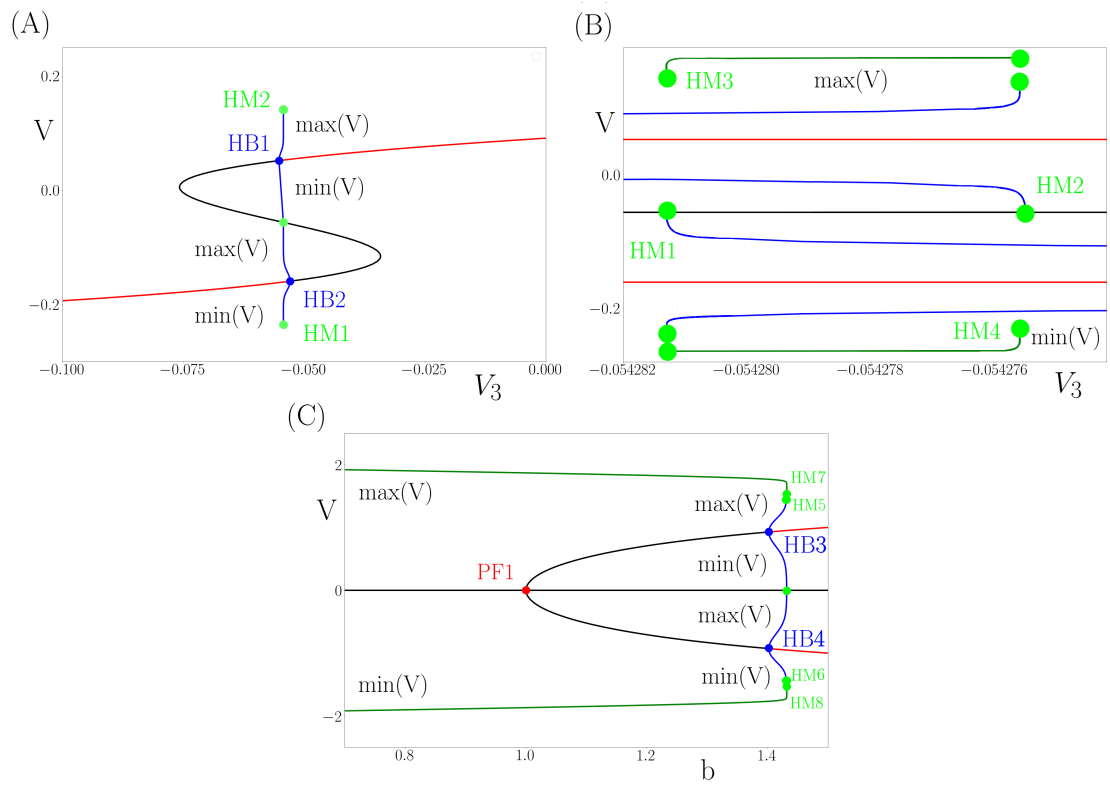

Figure 8.8: A: Morris-Lecar bifurcation diagram with respect to parameter V_3. The chosen solution measure for limit cycles is the max and min of V. Equilibria are represented by the solid black line (stable) and dashed black line (unstable). Limit cycles are represented in green (stable) and blue (unstable). Hopf bifurcation points are represented by black dots, and homoclinic bifurcations by red dots. B: Zoom on the periodic branches which correspond to eight-shaped solutions. Parameter values are: $I = -0.23395$ mA, $V_1 = 0.0$ V, $V_2 = 0.2$ V, $V_3 = -0.0542782$ V, $V_4 = 1.581236360973563$ V, $\tau = 10$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V. C: FitzHugh-Nagumo bifurcation diagram with respect to parameter b. Parameters values are: $I = 0.0$ mA, $a = 0$, $\varepsilon = 0.1$.

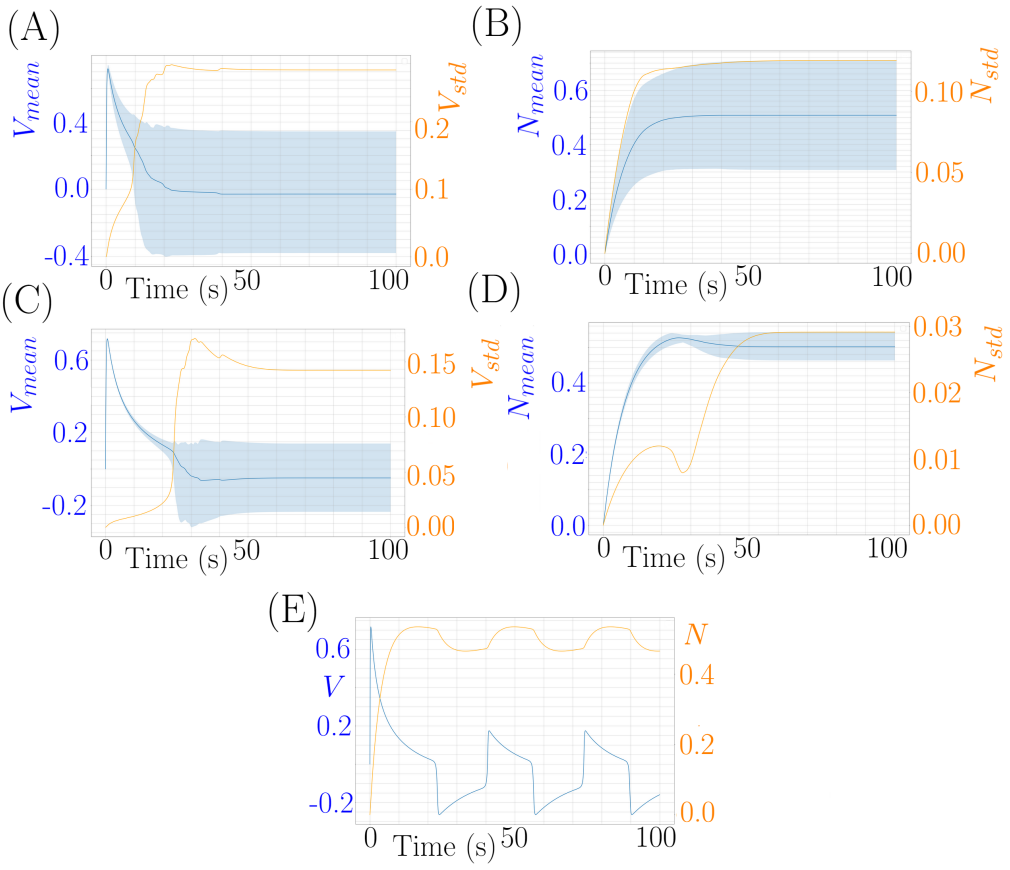

Figure 8.9: Sensitivity analysis for the Morris-Lecar system with respect to the two parameters V_3 and V_4 and with different range of uniform variation. A (V-component of the solution) and B (N-component): V_3 and V_4 have an uniform distribution of ± 1 around the parameter value. **C** (V-component) and **D** (N-component): V_3 and V_4 have an uniform variation of ± 0.1 around the parameter value. **E**: V_3 and V_4 without any variation of parameter. Parameters values are: $I = -0.23395$ mA, $V_1 = 0.0$ V, $V_2 = 0.2$ V, $V_3 = -0.0542782$ V, $V_4 = 1.581236360973563$ V, $\tau = 10$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V.

the circuit in order to bring out a resonator behavior on this integrator type neuron. Obviously, the basic Morris-Lecar model makes it possible to bring out this kind of behavior numerically. In Fig. 8.10, we present an attempt to obtain this resonator behavior, via tests of multiple values of I_0 in Chapter 3. What we notice is that we find the addition of spikes in the trains, but that no sub-threshold oscillation is visible. It is therefore impossible here to change the circuit behavior.

Then, we propose to try to apply the continuation of the rudimentary type on our electronic circuit (Fig. 8.11), and what comes out of it is that the algorithm has no difficulty in following the branch of stable stationary points, but at the time of passing to the branch of stationary point unstable, see even before passing the fold, the behavior becomes erratic. This behavior seems indeed under the influence of the oscillatory characteristics of the model, and thus does not make it possible to obtain the desired rudimentary continuation.

8.4 Discussion and conclusion

We were able to highlight the different possible dynamics of the system, compare them with the model, but also the characteristics that are specific to the circuit. These specific dynamics do not

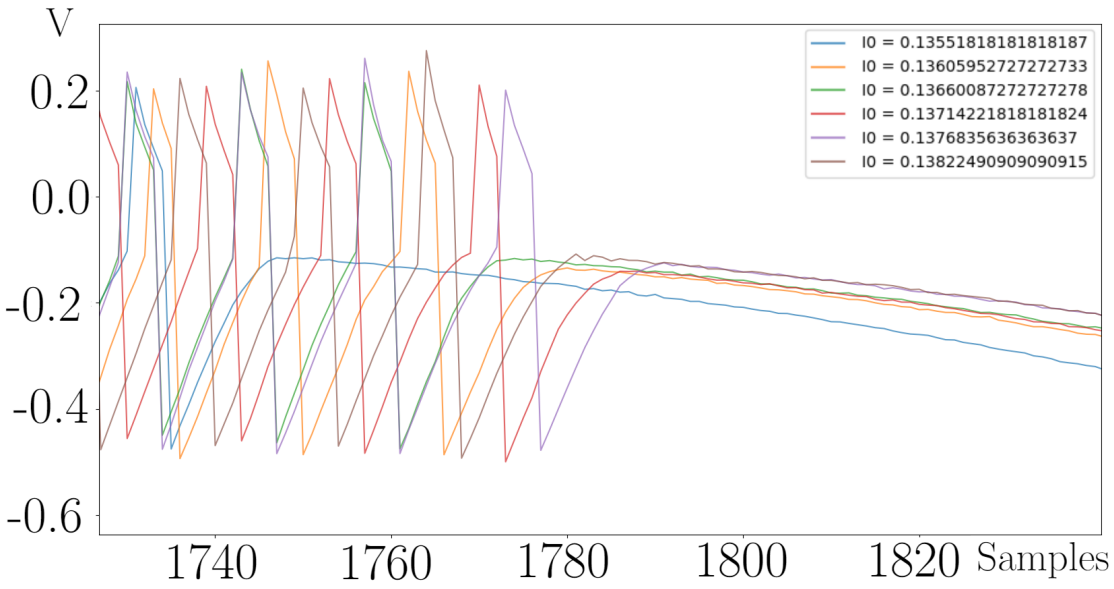

Figure 8.10: Integrator neuron (8.1) not acting as a resonator with the slow forced current. Parameter values are: $V_1 = 0.0$ V, $V_2 = 0.2$ V, $V_3 = -0.0542782$ V, $V_4 = 1.581236360973563$ V, $\tau = 10$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V, $J_0 = 1.8$, $\varepsilon = 0.2$, $\alpha = -3.5$, and I_0 is varying between 0.1354 and 0.1283.

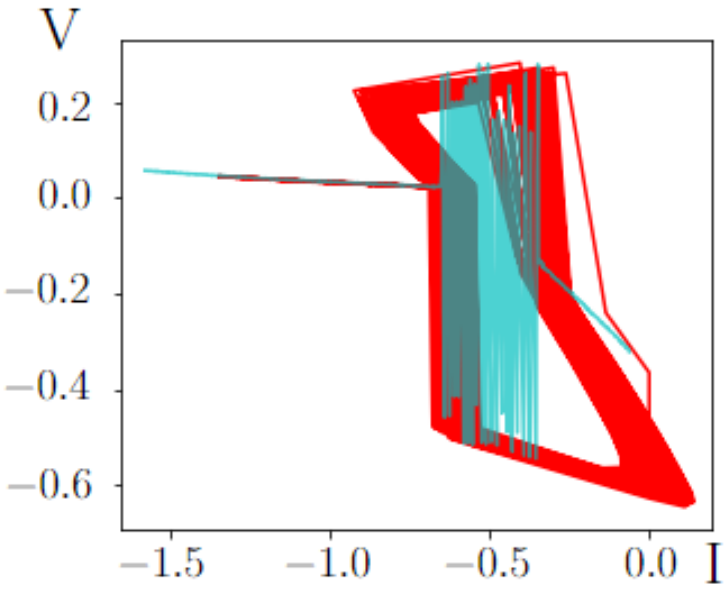

Figure 8.11: Morris-Lecar circuit current-clamp with slow ramp (blue curve) overlaid with the voltage-clamp with a slow ramp (red curve). Parameter values are: $V_1 = 0.0$ V, $V_2 = 0.2$ V, $V_3 = -0.0542782$ V, $V_4 = 1.581236360973563$ V, $\tau = 10$ ms, $C = 1.025$ µF, $g^*_{Ca} = 4.0$ mS, $V_{Ca} = 1.0$ V, $g^*_K = 8.0$ mS, $V_K = -0.66$ V, $g_L = 2.0$ mS, $V_L = 0.5$ V, $K = 3$, $\varepsilon = 0.2$

harm the study of the Morris-Lecar model, however for other reasons it was impossible for us to extract results which go in the direction of the transformation integrator to resonator, of the rudimentary continuation or of the CBCE.

In all cases, the main problem came from the possibility of interacting in ”real time” with the circuit. Indeed, during our experiments, it was possible to see that involving a control term in our

equations induced erratic solutions, this being in fact due to a delay between the reading of the information, and the sending. We obviously tried different software to interact with the circuit, namely LabView, Python or Matlab packages. Our main assumption about this delay would come from our way of converting the potential coming out of our acquisition device into current going into the circuit. This hypothesis is even reinforced by the fact we have been already being able to obtain a good rudimentary continuation curve on real neurons, so the problem is probably coming from the material.

In addition, it is important to note that not all of the parameters were adjustable via the computer insofar as certain parameters had to be modified physically on the circuit. Thus, we could not do a continuation on the whole set of parameters.

In conclusion, this study falls perfectly within the scope of the thesis, insofar as we experimentally study the behavior of a circuit which is supposed to have the behavior of a mathematical model of a single neuron. However, our experimental setup was limited for the study of the methods presented in this thesis.

Chapter 9

Appendix: Code scripts

In this chapter, all the codes used to complete this thesis are provided. The codes are sorted by chapter for better orientation.

9.1 Chapter 2: Integrator and resonator neurons

This first code is written for the software XPPAUT. It gives the 2-dimensional Morris-Lecar model, plus the 2-dimensional slow oscillator in order to obtain the integrator dynamic:

```
dV/dt = (I - gl*(V-El) - gna*Minf(V)*(V-Ena) - gk*N*(V - Ek))/C
dN/dt = (Ninf(V)-N)/taun
dI/dt = eps*(-J)
dJ/dt = eps*(I-I0)

# gating functions
Minf(V) = 1/(1+exp((Vmhalf-V)/Km))
Ninf(V) = 1/(1+exp((Vnhalf-V)/Kn))


# parameter values

par I0=-5.48
par C=1, taun=1
par El=-80
par Ena=60
par Ek=-90
par gl=8
par gna=20
par gk=10
par Vmhalf=-20
par Km=15
par Vnhalf=-25
par Kn=5
par eps=0.001

# initial conditions
init V=-76, N=3e-5, I=-5.43, J=10


# numerics
@ total=10000,dt=0.01
```

```
@ meth=rk4
@ xp=I,yp=V, xlo=-200,xhi=200,ylo=-120,yhi=0
@ bounds=9000000, maxstor=9000000
done
```

Then, we propose the same model, but with the feedback control included in the slow oscillator, in order to obtain a resonator behavior:

```
dV/dt = (I - gl*(V-El) - gna*Minf(V)*(V-Ena) - gk*N*(V - Ek))/C
dN/dt = (Ninf(V)-N)/taun
dI/dt = eps*(alpha*V-J)
dJ/dt = eps*(I-I0)

# gating functions
Minf(V) = 1/(1+exp((Vmhalf-V)/Km))
Ninf(V) = 1/(1+exp((Vnhalf-V)/Kn))


# parameter values

par I0=-4
par alpha=-4
par C=1, taun=1
par El=-80
par Ena=60
par Ek=-90
par gl=8
par gna=20
par gk=10
par Vmhalf=-20
par Km=15
par Vnhalf=-25
par Kn=5
par eps=0.001

# initial conditions
init V=-61.81, N=3e-5, I=-4, J=240.75


# numerics
@ total=5000,dt=0.01
@ meth=rk4
@ xp=I,yp=V, xlo=-200,xhi=200,ylo=-120,yhi=0
@ bounds=9000000, maxstor=9000000
done
```

9.2 Chapter 3: Immature neuron excitability

In this section, we propose to see first the XPPAUT code which gives the 3-states dynamicn with no flipper behavior:

```
# Model of hippocampal pyramidal neurons
```

```
# Ref.: Chizhov & Graham, PRE 75, 2007.
# reduced version

# ODEs
V' = (Iappl-INa-IDR-IA-IL-s*(V-Vus))/C
x1' = A31*(1-x1-x2)-x1*(A12+A13)
x2' = A12*x1-x2*A23
n' = ( ninf(V)-n )/taun( V)
yK' = (yKinf(V)-yK)/tauyK(V)
nA' = (nAinf(V)-nA)/taunA(V)
lA' = (lAinf(V)-lA)/taulA(V)


# Functions for equations of xi, i={1,2,3}
par A12=3.0
A13=1/(tau13min+1/exp((V-V13half)/k13))
A31=1/(tau31min+1/exp((V-V31half)/k31))
A23=1/(tau23min+1/(1/(tau23max-tau23min)+exp((V-V23half)/k23)))

# Additional parameters for the above functions
par tau13min=0.33333, V13half=-51.0, k13=-2.0
par tau31min=0.33333, V31half=-42.0, k31=1.0
par tau23min=1.0, V23half=-53.0, k23=-1.0, tau23max=100.0

# Activation & Inactivation functions
an(V)=0.17*exp((V+5)*0.09)
bn(V)=0.17*exp(-(V+5)*0.022)
taun(V)=1/(an(V)+bn(V))+0.8
ninf(V)=an(V)/(an(V)+bn(V))
tauyK(V)=300
yKinf(V)=1/(1+exp((V+68)*0.038))
anA(V)=0.08*exp((V+41)*0.089)
bnA(V)=0.08*exp(-(V+41)*0.016)
nAinf(V)=anA(V)/(anA(V)+bnA(V))
taunA(V)=1/(anA(V)+bnA(V))+1
alA(V)=0.04*exp(-(V+49)*0.11)
blA(V)=0.04
lAinf(V)=alA(V)/(alA(V)+blA(V))
taulA(V)=1/(alA(V)+blA(V))+2

# Currents
INa=gbarNa*x1*(V-VNa)
IDR=gbarDR*n*yK*(V-VK)
IA=gbarA*nA^4*lA^3*(V-VK)
IL=gbarL*(V-VL)
aux NaI=gbarNa*x1*(V-VNa)
aux DRI=gbarDR*n*yK*(V-VK)
aux AI=gbarA*nA^4*lA^3*(V-VK)
aux LI=gbarL*(V-VL)

# Reversal potentials and maximum conductances
```

```
par VNa=65.0, VK=-70.0, VL=-64.96
par gbarNa=2.28, gbarDR=0.76, gbarA=4.36, gbarL=0.048
par s=0.2, Vus=-60.0

# Initial Conditions
init V=-65, x1=0, x2=0, n=0.0012818, yK=0.46631
init nA=0.078559, lA=0.84545


# Fixed applied current
par Iappl=3.2

# Membrane capacitance
par C=0.70

# Initial Conditions
init V=-65, x1=0, x2=0

# Numerics
@ total=2000, method=rk4, dt=0.01
@ xhi=200,ylo=-80,yhi=40
@ maxstor=1000000
@ bounds=10000000

done
```

Then the 4-states case, so with the flipper phenomena:

```
# Model of hippocampal pyramidal neurons
# Ref.: Chizhov & Graham, PRE 75, 2007.

# ODEs
V' = (Iappl-INa-IDR-IA-IL-s*(V-Vus))/C
x1' = A21*x2+A31*x3+A41*(1-x1-x2-x3)-x1*(A12+A13+A14)
x2' = A12*x1+A32*x3+A42*(1-x1-x2-x3)-x2*(A21+A23+A24)
x3' = A13*x1+A23*x2+A43*(1-x1-x2-x3)-x3*(A31+A32+A34)
n' = (ninf(V)-n)/taun(V)
yK' = (yKinf(V)-yK)/tauyK(V)
nA' = (nAinf(V)-nA)/taunA(V)
lA' = (lAinf(V)-lA)/taulA(V)

# Fixed applied current
par Iappl=3.2

# Reversal potentials and maximum conductances
par gbarNa=2.28, VNa=65.0, VK=-70.0, VL=-64.96
par gbarDR=0.76, gbarA=4.36, gbarL=0.048
par s=0.2, Vus=-60.0

# Membrane capacitance
par C=0.70

```

```
# Parameters for equations of xi, i={1,2,3}
par A12=3.0, A21=0.0, A24=0.0, A32=0.0, A42=0.0, A43=0.0

# Functions for equations of xi, i={1,2,3}
A13=f113(V)
A14=f114(V)
A23=f223(V)
A31=f131(V)
A34=f234(V)
A41=f141(V)
f113(V)=1/(tau13min+1/exp((V-V13half)/k13))
f114(V)=1/(tau14min+1/exp((V-V14half)/k14))
f131(V)=1/(tau31min+1/exp((V-V31half)/k31))
f141(V)=1/(tau41min+1/exp((V-V41half)/k41))
f223(V)=1/(tau23min+1/(1/(tau23max-tau23min)+exp((V-V23half)/k23
    )))
f234(V)=1/(tau34min+1/(1/(tau34max-tau34min)+exp((V-V34half)/k34
    )))

# Additional parameters for the above functions
par tau13min=0.33333, V13half=-51, k13=-2
par tau14min=0.33333, V14half=-57, k14=-2
par tau23min=1, V23half=-53, k23=-1, tau23max=100
par tau31min=0.33333, V31half=-42, k31=1
par tau34min=1, V34half=-60, k34=-1, tau34max=100
par tau41min=0.33333, V41half=-51, k41=1

# Activation & Inactivation functions
an(V)=0.17*exp((V+5)*0.09)
bn(V)=0.17*exp(-(V+5)*0.022)
taun(V)=1/(an(V)+bn(V))+0.8
ninf(V)=an(V)/(an(V)+bn(V))
tauyK(V)=300
yKinf(V)=1/(1+exp((V+68)*0.038))
anA(V)=0.08*exp((V+41)*0.089)
bnA(V)=0.08*exp(-(V+41)*0.016)
nAinf(V)=anA(V)/(anA(V)+bnA(V))
taunA(V)=1/(anA(V)+bnA(V))+1
alA(V)=0.04*exp(-(V+49)*0.11)
blA(V)=0.04
lAinf(V)=alA(V)/(alA(V)+blA(V))
taulA(V)=1/(alA(V)+blA(V))+2

# Currents
INa=gbarNa*x1*(V-VNa)
IDR=gbarDR*n*yK*(V-VK)
IA=gbarA*nA^4*lA^3*(V-VK)
IL=gbarL*(V-VL)
aux NaI=gbarNa*x1*(V-VNa)
aux DRI=gbarDR*n*yK*(V-VK)
aux AI=gbarA*nA^4*lA^3*(V-VK)
```

```
aux LI=gbarL*(V-VL)

# Initial Conditions
init V=-65, x1=0, x2=0, x3=0, n=0.0012818, yK=0.46631
init nA=0.078559, lA=0.84545

# Numerics
@ total=2000, method=rk4, dt=0.01
@ xhi=200,ylo=-80,yhi=40
@ maxstor=1000000
@ bounds=10000000

done
```

9.3 Chapter 4: Rudimentary continuation

This XPPAUT code provides both at the same time, the current-clamp and the voltage-clamp protocol applied on a Morris-Lecar model. It is possible to overlay them by following this command: "Graphic stuff" -¿ "(A)dd curve" -¿ "X-axis: IAPPL, Y-axis: Vi".

```
# ODEs voltage clamp
V'=(-gL*(V-VL)-gK*w*(V-VK)-gCa*minf(V)*(V-VCa)+k*(V-Vtilde))/C
w'=eps*lamw(V)*(winf(V)-w)
Vtilde'=delta
# ODEs current clamp
Vi'=(-gL*(Vi-VL)-gK*wi*(Vi-VK)-gCa*minfi(Vi)*(Vi-VCa)+Iappl)/C
wi'=eps*lamwi(Vi)*(winfi(Vi)-wi)
Iappl'=deltai


aux IapplV=k*(V-Vtilde)

# Eq curve in (IapplV,V):
IapplV=gL*(V-VL)+gK*winf(V)*(V-VK)+gCa*minf(V)*(V-VCa)

# (in)activation functions
minf(V)=0.5*(1+tanh((V-V1)/V2))
winf(V)=0.5*(1+tanh((V-V3)/V4))
lamw(V)=cosh((V-V3)/(2*V4))
minfi(Vi)=0.5*(1+tanh((Vi-V1)/V2))
winfi(Vi)=0.5*(1+tanh((Vi-V3)/V4))
lamwi(Vi)=cosh((Vi-V3)/(2*V4))

# Parameters
par C=20
par gK=8.0
par gL=2.0
par eps=0.067
par V3=11.0
par V4=17.4
par VK=-74
```

```
par VL=-60
par VCa=80
par V1=-0.5
par V2=14.0
par k=-20.0

# ramp-up parameters
par delta=0.01, deltai=0.01, gCa=5.0

# Initial conditions (ramp-up)
init V=-99.0, w=5.1e-05, Vtilde=-100.0
init Vi=-99.0, wi=5.1e-05, Iappl=-40.0

# Numerics ramp up
@ total=12000.0
@ method=rk4, dt=0.005
@ xp=IapplV, yp=V, xlo=-10, xhi=150, ylo=-80, yhi=20
@ maxstor=900000
@ bounds=900000
done
```

9.4 Chapter 5: Continuation Based on Controled Experiments

9.4.1 Part 1: Fixed point case

In this part, the Matlab code is written to be used in interaction with a FitzHugh-Nagumo Simulink model which is not provided here. You can anyway make the choice to change the inputs and the parameters of the following code.

The main advantages of this Simulink model is that you can consider it as an external experiment from what you have to import the data to your workspace like a real experiment and that it is possible to add noise (Gaussian noise or band-limited white noise.)

For our experiments, we use the *band-limited white noise* block with different noise power level. This block generates normally distributed random numbers that are suitable to use in continuous system. For the integration of the signal, we use the *Integrator* block.

We propose now to give all the functions, we used during the process.

This first function allows to initialize the Simulink model with the good parameters. This function takes as input all the needed parameters, and doesn't have any output.

```
function X = INITIALISATION(A, B, C, D, E, EPS, K, V_TILDE, V0,
    W0, NOISE)
        set_param('FitzhughNagumo_circuit/noise ON OFF','Value',
            string(NOISE));
        set_param('FitzhughNagumo_circuit/a','Value', string(A))
            ;
        set_param('FitzhughNagumo_circuit/b','Value', string(B))
            ;
        set_param('FitzhughNagumo_circuit/c','Value', string(C))
            ;
        set_param('FitzhughNagumo_circuit/d','Value', string(D))
            ;
```

```
7           set_param('FitzhughNagumo_circuit/e','Value', string(E))
                ;
8           set_param('FitzhughNagumo_circuit/epsilon','Value',
                string(EPS));
9           set_param('FitzhughNagumo_circuit/gain','Value', string(
               K));
10          set_param('FitzhughNagumo_circuit/target','Value',
                string(V_TILDE));
11          set_param('FitzhughNagumo_circuit/v0','Value', string(V0
                ));
12          set_param('FitzhughNagumo_circuit/w0','Value', string(W0
                ));
13  end
```

The next function extract the data from Simulink.
The inputs are:

- Vtilde : the target for V,
- E : the bifurcation parameter,
- t :the pause time to extract the data from Simulink (obligated),
- range : the range of samples we want to extract from the end of real-time series in order to compute a mean value.

Outputs:

- V : the potential mean of the real-time time series,
- STD : the standard deviation of the studied range.

```
1   function [V, STD] = SIMULATION(Vtilde, E, t, range)
2           set_param('FitzhughNagumo_circuit/target','Value',
                string(Vtilde));
3           set_param('FitzhughNagumo_circuit/e','Value', string(E))
                ;
4           pause(t);
5           set_param('FitzhughNagumo_circuit', 'SimulationCommand',
                 'WriteDataLogs');
6           v_output = evalin('base','v_output');
7           [V, STD] = results_sim(v_output, range); %Extracting V
8   end
```

The next function compute the mean and the standard deviation of the recorded signal.
The inputs are:

- x_output : the recorded signal,
- range : the range of samples we want to extract from the end of real-time series in order to compute a mean value.

Outputs:

- X : the mean of the last time series values,

- Y : the standard deviation of the same time series last values.

```
function [X,Y] = results_sim(x_output, range)
        %X  = x_output.Data(length(x_output.Data));
        X  = mean(x_output.Data((length(x_output.Data)-range:
            length(x_output.Data))));
        Y  = std(x_output.Data((length(x_output.Data)-range:
            length(x_output.Data))));
end
```

The next function gives the residual computation.
Inputs:

- X_exp : the value extracted from the experiment,
- X_pred: the predicted value of the same variable.

Outputs:

- F : the computed residual.

```
function F = RESIDUAL(X_exp, X_pred)
        F = X_exp - X_pred;
end
```

The next function computes the prediction step.
Inputs:

- MODEL_guess : vector containing the guesses for V and E,
- MODEL_dv : the 2-dimensional direction vector,
- ds : the continuation step.

Outputs:

- MODEL_pred : vector containing the predictions for V and E.

```
function MODEL_pred = PREDICTION(MODEL_guess, MODEL_dv, ds)
        MODEL_pred = [MODEL_guess(1)+ds*MODEL_dv(1);
                      MODEL_guess(2)+ds*MODEL_dv(2)];
end
```

The next function computes the direction vector udpate with the secant method.
Inputs:

- MODEL_V : vector of all the calculated values with the algorithm for V,
- MODEL_L : the same type of vector but for the parameter E,
- p : the studied point,

Outputs:

- direct_v : the updated 2-dimensional direction vector,

```
function direct_v = UPDATE_SECANT(MODEL_V, MODEL_L, p)
    direct_v = [MODEL_V(p) - MODEL_V(p-1);
                MODEL_L(p) - MODEL_L(p-1)];
    direct_v = direct_v/norm(direct_v);
end
```

The next function computes the finite differences.
Inputs:

- F_REF : the residual based on the unperturbed experiments,
- F_PERT : the residual based on the perturbed case,
- H : the perturbation value.

Ouputs:

- Fx :the finite difference value.

```
function Fx = DERIVATIVE (F_REF, F_PERT, H)
        Fx = 1/H * (F_PERT - F_REF); % Fx(k) = 1/h * (F(x_tilde+
           h, lambda)-F(x_tilde , lambda))
end
```

The next function computes the direction vector udpate with the tangent method.
Inputs:

- Fx : the Jacobian matrix extracted from the Newton's or Broyden's method.

Outputs:

- direct_v : the updated direction vector.

```
function direct_v = UPDATE_TANGENT(Fx)
        vec = zeros(length(Fx),1);
        vec(length(Fx),1) = 1;
        direct_v = Fx\vec;
        direct_v = direct_v/ norm(direct_v);
end
```

The next function computes the Newton's iteration method.
Inputs:

- MODEL_pred : vector containing the predictions for V and E,
- MODEL_dv : the 2-dimensional direction vector,
- MODEL_guess : vector containing the guesses for V and E,
- t :the pause time to extract the data from Simulink (obligated),
- ITER : the number of iteration for the iteration method,
- range : the range of samples we want to extract from the end of real-time series in order to compute a mean value.

- H : the perturbation parameter,
- ds : the continuation step,
- tol : the tolerance threshold to break the iteration method.

Outputs:

- V_REF : the mean of the last time series values,
- V_STD : the standard deviation of the same time series last values,
- MODEL_pred : updated vector containing the predictions for V and E,
- Fx : the Jacobian matrix extracted from the Newton's method,
- Hx : the extended problem.

```
function [V_REF, V_STD, MODEL_pred, Fx, Hx] = NEWTON_METHOD(
    MODEL_pred, MODEL_dv, MODEL_guess, t, ITER, range, H, ds, tol
    )
     for iter=1:ITER

         %% V REFERENCE => F(Vtilde, lambda)
         [V_REF, V_STD] = SIMULATION(MODEL_pred(1), MODEL_pred(2)
             , t, range); % Simulating with new parameters

         %% V REFERENCE => F(Vtilde+H, lambda)
         [V_PERT_Vtilde, ~] = SIMULATION(MODEL_pred(1)+H,
             MODEL_pred(2), t, range); % Simulating with new
             parameters

         %% V REFERENCE => F(Vtilde, lambda+H)
         [V_PERT_L, ~] = SIMULATION(MODEL_pred(1), MODEL_pred(2)+
             H, t, range); % Simulating with new parameters

         %% RESIDUALS :
         F_REF = RESIDUAL(V_REF, MODEL_pred(1));
         F_PERT_Vtilde = RESIDUAL(V_PERT_Vtilde, MODEL_pred(1)+H)
             ;
         F_PERT_L = RESIDUAL(V_PERT_L, MODEL_pred(1));

         %% DERIVATIVES :
         Fx_Vtilde = DERIVATIVE(F_REF, F_PERT_Vtilde, H);
         Fx_L = DERIVATIVE(F_REF, F_PERT_L, H);
         Fx = [[Fx_Vtilde, Fx_L];
                 MODEL_dv.'];

         %% EXTENDED MATRIX :
         H_Vtilde = (MODEL_pred(1)-MODEL_guess(1))* MODEL_dv(1);
         H_L = (MODEL_pred(2)-MODEL_guess(2))* MODEL_dv(2);
         Hx = [F_REF;
                H_Vtilde + H_L - ds];
```

```
29
30          %% NEWTON :
31          MODEL_pred_previous = MODEL_pred;
32          MODEL_pred = MODEL_pred - (Fx\Hx);
33
34          %% CONDITION TO BREAK :
35          if norm(Hx)<tol && norm(MODEL_pred - MODEL_pred_previous
                )<tol
36              % Checking ball size :
37              break
38          end % For the condition of break
39      end
40
41  end
```

The next function computes the Broyden's iteration method.
Inputs:

- MODEL_pred : vector containing the predictions for V and E,
- MODEL_dv : the 2-dimensional direction vector,
- MODEL_guess : vector containing the guesses for V and E,
- t :the pause time to extract the data from Simulink (obligated),
- ITER : the number of iteration for the iteration method,
- range : the range of samples we want to extract from the end of real-time series in order to compute a mean value.
- H : the perturbation parameter,
- ds : the continuation step,
- tol : the tolerance threshold to break the iteration method.

Outputs:

- V_REF : the mean of the last time series values,
- V_STD : the standard deviation of the same time series last values,
- MODEL_pred : updated vector containing the predictions for V and E,
- Fx : the Jacobian matrix extracted from the Newton's method,

```
1   function [V_REF, V_STD, MODEL_pred, Fx] = BROYDEN_METHOD(
        MODEL_pred, MODEL_dv, MODEL_guess, t, ITER, range, H, ds, tol
        )
2       for iter=1:ITER
3           if iter == 1
4               [V_REF, V_STD, MODEL_pred, Fx, Hx] = NEWTON_METHOD(
                    MODEL_pred, MODEL_dv, MODEL_guess, t, 1, range, H
                    , ds, tol);
5           else
```

```
            %% V REFERENCE => F(Vtilde, lambda)
            [V_REF, V_STD] = SIMULATION(MODEL_pred(1),
                MODEL_pred(2), t, range); % Simulating with new
                parameters

            %% RESIDUALS :
            F_REF = RESIDUAL(V_REF, MODEL_pred(1));

            %% EXTENDED MATRIX :
            H_Vtilde = (MODEL_pred(1)-MODEL_guess(1))* MODEL_dv
                (1);
            H_L = (MODEL_pred(2)-MODEL_guess(2))* MODEL_dv(2);
            Hx = [F_REF;
                  H_Vtilde + H_L - ds];

            %% DERIVATIVES :
            dFn = Hx - Hx_PREVIOUS;
            dXn = MODEL_pred - MODEL_pred_PREVIOUS;
            Fx = Fx_PREVIOUS + ((dFn-Fx_PREVIOUS*dXn)/norm(dXn)
                ^2)*dXn.';

        end

        [U,S,V] = svd(Fx);
        Fx = U*(S+1e-04)*V';

        if any(isnan(Fx),'all') || rank(Fx)<min(Fx, [], 'all')
            [V_REF, V_STD, MODEL_pred, Fx, Hx] = NEWTON_METHOD(
                MODEL_pred, MODEL_dv, MODEL_guess, t, 1, range, H
                , ds, tol);
        end

        %% NEWTON :
        MODEL_pred_PREVIOUS = MODEL_pred;
        Fx_PREVIOUS = Fx;
        Hx_PREVIOUS = Hx;
        MODEL_pred = MODEL_pred - (Fx\Hx);

        %% CONDITION TO BREAK :
        if norm(Hx)<tol && norm(MODEL_pred - MODEL_pred_PREVIOUS
            )<tol
            break
        end % For the condition of break
    end

end
```

The next code is the main script to launch the experiment:

```
close all
clear all
format longg
```

```
4
5  % DEBUG PARAMETER :
6  % 0 : No Debug Mode / 1 : Debug mode.
7  DEBUG = 0;
8  SHOW_CONV = 1; % Showing Convex Hull.
9  SHOW_STD = 0; % Showing errors.
10 SHOW_VARIATION = 0; % Showing errors.
11 % TIME OF PROCESS :
12 tic
13
14 %% SIMULATION PARAMETERS :
15 % The equations are :
16 % y'(1) = -y(1) *(-a+y(1))*(-b+y(1))-y(2)+K*(y(1)-vtilde)
17 % y'(2) = eps *(c*y(1)+d*y(2)+e)
18
19 A = 1.8;
20 B = 0.0;
21 C = 1.0;
22 D = -1.0;
23 EPS = 0.6;
24 K = -2;                    % Gain for the control
25 V0 = 0.;
26 W0 = 0.;
27
28 %% NOISE :
29 % For the Simulink experiment : Noise power : [3e-04]
30 % If Noise == 1 : ON, If Noise == 0 : OFF,
31 NOISE = 1;
32 Noise_Power = 3e-04;
33
34 %% CONTINUATION PARAMETERS :
35
36 VECTOR_TYPE = "Secant";          % direction vector type : "
   Tangent", "Secant"
37 ITERATION_METHOD = "BROYDEN";    % "NEWTON" OR "BROYDEN"
38 t = 0.1;                         % Pause time : 0.2 / 0.02
39 range = 200;                     % Size of the sample at the
   asymptotic solution : 250 / 25 noise
40 H = 0.001;                       % Derivative parameter
41 tol = 1e-05;                     % Tolerance to break the loop
42 Nb_point_cloud = 3;              % Number of point computed in
   order to make the mean position.
43 eps_circle = 0.05;                % Small parameter for the trust
    region around the last computed point. eps_circle = 0.5.
44
45 if VECTOR_TYPE == "Tangent"
46     V_TILDE = -1;           % Starting value for continuation
47     E = -1.5;                % Continuation parameter lambda
48     ds = 0.05;              % Step for the continuation : 0.005
49     points = 50;            % Nb of points for continuation : 400
50     ITER = 75;              % Nb of iteration for Newton : 20
```

```
51 elseif VECTOR_TYPE == "Secant"
52     V_TILDE = 3;            % Starting value for continuation
53     E = -1.2;                 % Continuation parameter lambda
54     ds = 0.05;              % Step for the continuation : 0.005
55     points = 50;             % Nb of points for continuation : 400
56     ITER = 75;               % Nb of iteration for Newton : 20
57 end
58
59 %% INITIALIZE THE SIMULINK MODEL :
60 % Before to apply the code on the Simulink model, we initialize
       it with the
61 % basic parameters.
62 INITIALISATION(A, B, C, D, E, EPS, K, V_TILDE, V0, W0, NOISE);
63 pause(1)
64
65 %% Initialisation of the modelling continuation
66 MODEL_guess = [V_TILDE; E];          % First guess
67 MODEL_dv = [0;1];                    % Tangent direction
68 MODEL_dv_stock = MODEL_dv;           % Stock tangent directions
       for plot.
69 MODEL_V = [V_TILDE];                 % Saving V data
70 MODEL_STD = [0];                     % Saving Standard deviation
       of the range V.
71 MODEL_VARIATION = [0;0];             % Saving the standard
       deviation of the computed points.
72 MODEL_CONV = [];                     % Saving points for Convex
       Hull
73 MODEL_Vtilde = [MODEL_guess(1)];     % Saving Vtilde data
74 MODEL_L = [MODEL_guess(2)];          % Saving L data
75 MODEL_STABILITY = [0];               % Saving stability of the
       point
76 MODEL_NATURE = [0];                  % Saving nature of the point
77
78 for p=2:points % Loop for the number of points, we want to
       compute.
79
80     V_mean = []; % Empty list of the Nb_point_cloud V_REF values
           computed for the new point.
81     STANDARD_mean = []; % Empty list of the Nb_point_cloud
           Standard deviation values computed for the new point.
82     MODEL_pred_mean = []; % Empty list of the Nb_point_cloud
           Vtilde and Lambda coordinates computed for the new point.
83     MODEL_Fx_mean = [[0, 0]; [0, 0]]; % Saving mean for Jacobian
           matrix for stability
84
85     %% PREDICTION STEP
86     MODEL_pred = PREDICTION(MODEL_guess, MODEL_dv, ds);     %
           PREDICTION
87
88     for cloud=1:Nb_point_cloud % Loop for the number of try we
           want to compute for one new points, then making the mean.
```

```
89          if ITERATION_METHOD == "NEWTON"
90              [V_REF, V_STD, MODEL_pred, Fx, Hx] = NEWTON_METHOD(
                    MODEL_pred, MODEL_dv, MODEL_guess, t, ITER, range
                    , H, ds, tol);
91          elseif ITERATION_METHOD == "BROYDEN"
92              [V_REF, V_STD, MODEL_pred, Fx] = BROYDEN_METHOD(
                    MODEL_pred, MODEL_dv, MODEL_guess, t, ITER, range
                    , H, ds, tol);
93          end
94
95          %% Trust Region computation :
96          % We want to check if the new computed point is
                contained in the
97          % circle centered on the previous computed point. So we
                compute
98          % (x-a)^2 - (y-b)^2 < R^2. For R, we fix it at R=2 for
                the first
99          % computed point, then R will be equal to the distance
                between the
100         % two last points plus a smal parameter for the
                variation of
101         % distance between two computed points.
102
103         if p==2
104             R = 2;
105         else
106             R = sqrt((MODEL_V(p-1)-MODEL_V(p-2))^2 + (MODEL_L(p
                    -1)-MODEL_L(p-2))^2)+eps_circle;
107         end
108         if sqrt((MODEL_pred(1)-MODEL_V(p-1))^2 + (MODEL_pred(2)-
                MODEL_L(p-1))^2)<R
109             V_mean = [V_mean, V_REF];
110             STANDARD_mean = [STANDARD_mean, V_STD];
111             MODEL_pred_mean = [MODEL_pred_mean, MODEL_pred];
112             MODEL_Fx_mean = MODEL_Fx_mean + Fx;
113         else
114             V_mean = [V_mean, NaN];
115             STANDARD_mean = [STANDARD_mean, NaN];
116             MODEL_pred_mean = [MODEL_pred_mean, [NaN;NaN]];
117         end
118     end
119
120     %% STABILITY :
121
122     Fx = MODEL_Fx_mean / Nb_point_cloud ;
123     [Stability, Nature] = STABILITY(Fx);
124     MODEL_STABILITY = [MODEL_STABILITY, Stability];
125     MODEL_NATURE = [MODEL_NATURE, Nature];
126
127     %% ALLOCATING :
```

```
128     % Here, we compute the mean of the Nb_point_cloud tries
            computed for the new
129     % point and we allocate the data in our lists.
130
131     % Mean of the different trials :
132     V_REF = nanmean(V_mean);
133     V_STD = nanmean(STANDARD_mean);
134     MODEL_pred = nanmean(MODEL_pred_mean,2);
135
136     % For the uncertainty :
137     MODEL_CONV = [MODEL_CONV; [MODEL_pred_mean]];
138     MODEL_VARIATION = [MODEL_VARIATION, nanstd(MODEL_pred_mean,
            0, 2)];
139     MODEL_STD = [MODEL_STD, V_STD];
140
141     % Allocating the results for the iterations :
142     MODEL_V = [MODEL_V, V_REF];
143     MODEL_Vtilde = [MODEL_Vtilde, MODEL_pred(1)];
144     MODEL_L = [MODEL_L, MODEL_pred(2)];
145     MODEL_guess = MODEL_pred;
                                                          %NEW
            GUESSES
146
147     %% VECTOR UPDATE :
148     if VECTOR_TYPE == "Tangent"
149         MODEL_dv = UPDATE_TANGENT(Fx);
                                                              %NEW
                GUESSES
150     elseif VECTOR_TYPE == "Secant"
151         MODEL_dv = UPDATE_SECANT(MODEL_V, MODEL_L, p); % To
                update the secant, we just have to take the two last
                computed points: [x(p) - x(p-1); y(p) - y(p-1)]
152     end
153
154     MODEL_dv_stock = [MODEL_dv_stock, MODEL_dv];
155
156 end
157
158 %% ####################
159 %% ODE45 CONTINUATION :
160 %% ####################
161 % To compare our results we compute the Tangent continuation,
        with the
162 % Newton's method, on the equations.
163 %% SIMULATION PARAMETERS :
164 A = 1.8;
165 B = 0.0;
166 C = 1.0;
167 D = -1.0;
168 E = -1.0;
169 EPS = 0.6;
```

```
K = -2;                    % Gain for the control
V_TILDE = -1;              % Starting value for continuation
V0 = 0.;
W0 = 0.;

%% CONTINUATION PARAMETERS :
ds = 0.05;                % Step for the continuation : 0.005
H = 0.001;                 % Derivative parameter
tol = 1e-04;              % Tolerance to break the loop
points = 40;               % Nb of points for continuation : 400
ITER = 100;                 % Nb of iteration for Newton : 20
range = 250;                % Size of the sample at the asymptotic solution : 250 / 25 noise
t = 0.2;                   % Pause time : 0.2 / 0.02

[RESULTS, TS, EQ_FX] = ODE45_Continuation(A, B, C, D, E, EPS, K, V_TILDE, V0, W0, ds, H, tol, points, ITER, range);
EQ_V = RESULTS(1,:);
EQ_L = RESULTS(2,:);

%% ###################
%% PLOTTING RESULTS :
%% ###################

fprintf("\n")

figure
hold on
xlabel('E')
ylabel('V')
title("SIMULINK : Bifurcation diagram (Jan's Continuation)")
p1 = plot(MODEL_L(3:end), MODEL_V(3:end), '--r*');
p2 = plot(EQ_L(3:end), EQ_V(3:end), 'b--o');
p2.Color(4)= .02;
if SHOW_CONV == 1
    for CONV = [1:2:length(MODEL_CONV)]
        TOP = MODEL_CONV(CONV,:);
        TOP(isnan(TOP)) = nanmean(TOP);
        BOT = MODEL_CONV(CONV+1,:);
        BOT(isnan(BOT)) = nanmean(BOT);
        P = [TOP;BOT].';
        [k,av] = convhull(P);
        plot(P(:,2),P(:,1),'*k')
        hold on
        plot(P(k,2),P(k,1),'--k')
    end
end
if SHOW_VARIATION == 1
    xpos = MODEL_VARIATION(2,:);
    xneg = -MODEL_VARIATION(2,:);
    ypos = MODEL_VARIATION(1,:);
```

```
219     yneg = -MODEL_VARIATION(1,:);
220     errorbar(MODEL_L(3:end), MODEL_V(3:end),yneg(3:end),ypos(3:
            end),xneg(3:end),xpos(3:end),'ok')
221 end
222 if SHOW_STD == 1
223     errorbar(MODEL_L(3:end), MODEL_V(3:end),MODEL_STD(3:end),'ok
            ','horizontal')
224 end
225 legend("Simulink "+VECTOR_TYPE+" Continuation", "ODE45 Tangent
        Continuation");
226 hold off
227
228 figure
229 hold on
230 xlabel('E')
231 ylabel('V')
232 title("SIMULINK : Bifurcation diagram (Jan's Continuation)")
233 ylim([-1 2])
234 xlim([-0.24 -0.06])
235 p3 = plot(MODEL_L(2:end), MODEL_V(2:end), '--r*');
236 p4 = plot(EQ_L(3:end), EQ_V(3:end), 'b--o');
237 p4.Color(4)= .02;
238 if SHOW_CONV == 1
239     for CONV = [1:2:length(MODEL_CONV)]
240         TOP = MODEL_CONV(CONV,:);
241         TOP(isnan(TOP)) = nanmean(TOP);
242         BOT = MODEL_CONV(CONV+1,:);
243         BOT(isnan(BOT)) = nanmean(BOT);
244         P = [TOP;BOT].';
245         [k,av] = convhull(P);
246         plot(P(:,2),P(:,1),'*k')
247         hold on
248         plot(P(k,2),P(k,1),'--k')
249     end
250 end
251 if SHOW_VARIATION == 1
252     xpos = MODEL_VARIATION(2,:);
253     xneg = -MODEL_VARIATION(2,:);
254     ypos = MODEL_VARIATION(1,:);
255     yneg = -MODEL_VARIATION(1,:);
256     errorbar(MODEL_L(3:end), MODEL_V(3:end),yneg(3:end),ypos(3:
            end),xneg(3:end),xpos(3:end),'ok')
257 end
258 if SHOW_STD == 1
259     errorbar(MODEL_L(3:end), MODEL_V(3:end),MODEL_STD(3:end),'ok
            ','horizontal')
260 end
261 legend("Simulink "+VECTOR_TYPE+" Continuation", "ODE45 Tangent
        Continuation");
262 hold off
263
```

```
figure
hold on
xlabel('E')
ylabel('V')
title("SIMULINK : Bifurcation diagram (Jan's Continuation)")
p1 = plot(MODEL_L(3:end), MODEL_V(3:end), '--r*');
for i=3:length(MODEL_L)
    if MODEL_STABILITY(i) == 0 % Stable
        p2 = scatter(MODEL_L(i), MODEL_V(i), "red", "filled");
    elseif MODEL_STABILITY(i) == 1 % Unstable
        p2 = scatter(MODEL_L(i), MODEL_V(i), "black", "filled");
    elseif MODEL_STABILITY(i) == 2 % Neutral
        p2 = scatter(MODEL_L(i), MODEL_V(i), "blue", "filled");
    end
end
legend("Simulink "+VECTOR_TYPE+" Continuation", "Stable point",
   "Untable point");
hold off

% TIME PROCESS :
TimeElapsed = toc
```

We also propose a version of the code based on the Matlab ODE45 function, i.e. a differential equation solving method.

This method is also based on the FitzHugh Nagumo model as follow:

```
function dydt = equations(t, y, a, b, c, d, e, eps, K, vtilde)
        dydt = [-y(1) *(-a+y(1))*(-b+y(1))-y(2)+K*(y(1)-vtilde);
                eps *(c*y(1)+d*y(2)+e)];
end
```

And this is the main script:

```
function [X,Y, EQ_FX] = ODE45_Continuation(A, B, C, D, E, EPS, K
   , V_TILDE, V0, W0, ds, H, tol, points, ITER, range)

    % accuracy of the discretisation of the ODE
    options=odeset('RelTol',1e-08,'AbsTol',1e-08);

    %% PARAMETER INITIALIZATION
    y0 = [V0, W0];
    tspan = 0:0.1:30;

    %% Initialisation of the modelling continuation
    MODEL_guess = [V_TILDE; E];      % First guess
    MODEL_dv = [0;1];                % Tangent direction
    MODEL_V = [V0];      % Saving V data
    MODEL_Vtilde = [MODEL_guess(1)];      % Saving Vtilde data
    MODEL_L = [MODEL_guess(2)];      % Saving L data
    PLOT_TS_V = [];
    PLOT_TS_W = [];
    EQ_FX = [];

```

```
20      for p=2:points
21
22          %% CONTINUATION
23          % Prediction :
24          MODEL_pred = PREDICTION(MODEL_guess, MODEL_dv, ds);     %
                PREDICTION
25
26          for iter=1:ITER
27
28              %% V REFERENCE => F(Vtilde, lambda)
29              % dydt = equations(t, y, a, b, c, d, e, eps, K,
                    vtilde)
30              [t,y] = ode45(@(t,y) equations(t, y, A, B, C, D,
                    MODEL_pred(2), EPS, K, MODEL_pred(1)), tspan, y0)
                    ;
31              V_REF = mean(y(length(y)-range:length(y),1));
32              y0 = [y(length(y),1), y(length(y),2)];
33
34              PLOT_TS_V = [PLOT_TS_V; y(length(y),1)];
35              PLOT_TS_W = [PLOT_TS_W; y(length(y),2)];
36
37              %% V REFERENCE => F(Vtilde+H, lambda)
38              [t,y] = ode45(@(t,y) equations(t, y, A, B, C, D,
                    MODEL_pred(2), EPS, K, MODEL_pred(1)+H), tspan,
                    y0);
39              V_PERT_Vtilde = mean(y(length(y)-range:length(y),1))
                    ;
40              y0 = [y(length(y),1), y(length(y),2)];
41
42              %% V REFERENCE => F(Vtilde, lambda+H)
43              [t,y] = ode45(@(t,y) equations(t, y, A, B, C, D,
                    MODEL_pred(2)+H, EPS, K, MODEL_pred(1)), tspan,
                    y0);
44              V_PERT_L = mean(y(length(y)-range:length(y),1));
45              y0 = [y(length(y),1), y(length(y),2)];
46
47              %% RESIDUALS :
48              F_REF = RESIDUAL(V_REF, MODEL_pred(1));
49              F_PERT_Vtilde = RESIDUAL(V_PERT_Vtilde, MODEL_pred
                    (1)+H);
50              F_PERT_L = RESIDUAL(V_PERT_L, MODEL_pred(1));
51
52              %% DERIVATIVES :
53              Fx_Vtilde = DERIVATIVE(F_REF, F_PERT_Vtilde, H);
54              Fx_L = DERIVATIVE(F_REF, F_PERT_L, H);
55              Fx = [[Fx_Vtilde, Fx_L];
56                    MODEL_dv.'];
57              EQ_FX = [EQ_FX, norm(Fx)];
58
59              %% EXTENDED MATRIX :
```

```
60              H_Vtilde = (MODEL_pred(1)-MODEL_guess(1))* MODEL_dv
                    (1);
61              H_L = (MODEL_pred(2)-MODEL_guess(2))* MODEL_dv(2);
62              Hx = [F_REF;
63                    H_Vtilde + H_L - ds];
64
65              %% NEWTON :
66              MODEL_pred_previous = MODEL_pred;
67              MODEL_pred = MODEL_pred - (Fx\Hx);
68
69              %% CONDITION TO BREAK :
70              if norm(Hx)<tol && norm(MODEL_pred -
                    MODEL_pred_previous)<tol
71                  break
72              end % For the condition of break
73
74          end
75
76          %% ALLOCATING :
77          MODEL_V = [MODEL_V, V_REF];
78          MODEL_Vtilde = [MODEL_Vtilde, MODEL_pred(1)];
79          MODEL_L = [MODEL_L, MODEL_pred(2)];
80          MODEL_guess = MODEL_pred;
                                                                %NEW
                GUESSES
81
82          %% TANGENT VECTOR :
83          MODEL_dv = UPDATE_TANGENT(Fx);
84      end
85
86      X = [MODEL_V; MODEL_L];
87      Y = [PLOT_TS_V, PLOT_TS_W];
88
89  end
```

9.4.2 Part 2: Periodic case

In this section, we will provide the code which made it possible to code the continuation on the periodic branch of the FHN model simulated with the ODE45 function of Matlab. Some functions are the same as those in the part on stationary points: these will not be given again in this part and it is therefore enough to copy them from this part to make the code work.

In this code, we are interested in a continuation based on $q = 2$ calculated Fourier modes, and the studied model is a FitzHugh Nagumo model.

Inputs:

- t : the time used for ODE45,
- s : the solution computed for ODE45,
- a, b, c, d ,e : FHN parameters,
- x0, x1, x_1, x2, x_2 : the 5 Fourier coefficients,

- T : the period,
- K : the gain of the control,

Outputs:

- dydt : the solution after simulation of the model,

```
function dydt = FHN_ODE(t, s, eps, a, b, c, d, e, x0, x1, x_1,
    x2, x_2, T, K)

    x = s(1);
    y = s(2);
    dydt = zeros(2,1);

    if K ~= 0
        x_tilde = sqrt(1/T) * x0;
        x_tilde = x_tilde + (x1 * sqrt(2/T) * sin( (2 * pi/T) *
            t));
        x_tilde = x_tilde + (x_1 * sqrt(2/T) * cos(- (2 * pi/T)
            * t));
        x_tilde = x_tilde + (x2 * sqrt(2/T) * sin( (2 * pi/T) *
            2 * t));
        x_tilde = x_tilde + (x_2 * sqrt(2/T) * cos(- (2 * pi/T)
            * 2 * t));
    else
        x_tilde = 0;
    end

    dydt(1) = -x*(x-a)*(x-b) - y + K * (x-x_tilde);
    dydt(2) = eps * (c*x + d*y + e);

end
```

The next function computes the Newton's iteration method for the periodic case.
Inputs:

- MODEL_pred : vector containing the predictions for V and E,
- MODEL_dv : the 2-dimensional direction vector,
- MODEL_guess : vector containing the guesses for V and E,
- deriv_coeffs : the Fourier coefficients of the previous computed point derivative,
- ITER : the number of iteration for the iteration method,
- H : the perturbation parameter,
- ds : the continuation step,
- tol : the tolerance threshold to break the iteration method,
- K : the gain of the control,
- y0 : the initial conditions for ODE45,

Outputs:

- MODEL_pred : updated vector containing the predictions for the coefficients x_0, x_1, x_{-1}, x_2, x_{-2}, T and E,
- Fx : the Jacobian matrix extracted from the Newton's method,
- Hx : the extended problem.

```
function [MODEL_pred, Fx, Hx] = NEWTON_METHOD_PERIOD(MODEL_pred,
    MODEL_dv, MODEL_guess, deriv_coeffs, ITER, H, ds, tol, K, y0
    )

    global tspan
    %global y0
    global nb_samples
    global DEBUG
    global options

    global A
    global B
    global C
    global D
    global EPS

    for iter=1:ITER

        alloc = num2cell(MODEL_pred);
        [x0, x1, x_1, x2, x_2, T, lambda] = alloc{:};

        %% REFERENCE SIMULATION :
        % Obtaining the normalized coefficients from a reference
            simulation.
        sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D, lambda
            , x0, x1, x_1, x2, x_2, T, K), tspan, y0);
        norm_signal = deval(sol, (linspace(tspan(2)-T, tspan(2),
            nb_samples)), 1);
        norm_coeffs = MY_NORM_COEFFS(norm_signal,linspace(tspan
            (2)-T, tspan(2), nb_samples));     % NORMALIZED
            COEFFICIENTS
        Fref = norm_coeffs - MODEL_pred(1:5);

        %% EXTENDED PROBLEM FORMATION :
        % Residuals :
        Hx = Fref;
        % Phase condition :
        Hx = [Hx; norm_coeffs.' * deriv_coeffs];
        % Pseudo-arclength rule :
        Hx = [Hx; ((MODEL_pred-MODEL_guess).' * MODEL_dv) - ds];

        %% DIFFERENCES JACOBIAN TYPE :
```

```
36          Fx = DIFF_JAC(MODEL_pred, MODEL_dv, Fref, deriv_coeffs,
                H, K, y0);
37
38          %% NEWTON :
39          MODEL_pred_previous = MODEL_pred;
40          MODEL_pred = MODEL_pred - (Fx\Hx);
41
42          %% CONDITION TO BREAK :
43          if norm(Hx)<tol && norm(MODEL_pred - MODEL_pred_previous
                )<tol
44              break
45          end
46      end
47
48  end
```

The next function is computing the derivative of a signal, and then, the normalized Fourier coefficients of this derivative for $q = 2$ modes.

Inputs:

- signal : the time series,
- time : the corresponding time.

Outputs:

- deriv_coeffs : the coefficients of the derivative solution.

```
1   function deriv_coeffs = MY_DERIVATIVE_NORM_SIGNAL(signal,time)
2
3       time = time - time(1);  % shift time interval to start at 0:
            [0,T]
4
5
6       derivsignal=zeros(1,length(time)-1);
7       for i=1:length(derivsignal)
8       derivsignal(1,i)=(signal(1,i+1)-signal(1,i))/(time(1,i+1)-
            time(1,i));
9       end
10      %
11      time2=time(1:end-1);
12      %
13      deriv_coeffs = MY_NORM_COEFFS(derivsignal, time2); %
            NORMALIZED COEFFICIENTS
14
15  end
```

The next function is computing the normalized Fourier coefficients of a signal for $q = 2$ modes.

Inputs:

- signal : the time series,
- time : the corresponding time.

Outputs:

- coeffs : the normalized Fourier coefficients of a solution.

```
function coeffs = MY_NORM_COEFFS(signal, time)


    time = time - time(1); % shift time interval to start at 0: [0,T]
    T    = time(end);

    % For mode = 0 :
    F0  = sqrt(1/T) * signal ;
    FS0 = trapz(time, F0);

    % For mode = 1 :
    F1  = sqrt(2/T) * signal .*  sin((2*pi/T) * time);
    FS1 = trapz(time, F1);

    % For mode = -1 :
    F_1  = sqrt(2/T) * signal .*  cos((-2*pi/T) * time);
    FS_1 = trapz(time, F_1);

    % For mode = 2 :
    F2  = sqrt(2/T) * signal .*  sin((2*pi/T) * 2 * time);
    FS2 = trapz(time, F2);

    % For mode = -2 :
    F_2  = sqrt(2/T) * signal .*  cos((-2*pi/T) * 2 * time);
    FS_2 = trapz(time, F_2);

    % Fourier coefficients
    coeffs = [FS0; FS1; FS_1; FS2; FS_2];

end
```

The next function computes the Jacobian matrix for the Newton's problem thanks to ODE45.
Inputs:

- MODEL_pred : vector containing the predictions for V and E,
- MODEL_dv : the 2-dimensional direction vector,
- Fref : the residual obtained from the experiment without any parameter perturbation,
- deriv_coeffs : the Fourier coefficients of the previous computed point derivative,
- H : the perturbation parameter,
- K : the gain of the control,
- y0 : the initial conditions for ODE45,

Outputs:

- Fx : the Jacobian matrix extracted for the Newton's method,

```
function Fx  = DIFF_JAC(MODEL_pred, MODEL_dv, Fref, deriv_coeffs
    , H, K, y0)

    global tspan
    %global y0
    global nb_samples

    global A
    global B
    global C
    global D
    global EPS

    Fx = zeros(length(MODEL_pred));
    alloc = num2cell(MODEL_pred);
    [x0, x1, x_1, x2, x_2, T, lambda] = alloc{:};

    %% PERTURBATION ON COEFFS SIMULATION :
    for j=1:7

        if j == 1
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0+H, x1, x_1, x2, x_2, T, K), tspan, y0)
                ;
            Pert = [H; 0; 0; 0; 0];
        elseif j == 2
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0, x1+H, x_1, x2, x_2, T, K), tspan, y0)
                ;
            Pert = [0; H; 0; 0; 0];
        elseif j == 3
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0, x1, x_1+H, x2, x_2, T, K), tspan, y0)
                ;
            Pert = [0; 0; H; 0; 0];
        elseif j == 4
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0, x1, x_1, x2+H, x_2, T, K), tspan, y0)
                ;
            Pert = [0; 0; 0; H; 0];
        elseif j == 5
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0, x1, x_1, x2, x_2+H, T, K), tspan, y0)
                ;
            Pert = [0; 0; 0; 0; H];
        elseif j == 6
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda, x0, x1, x_1, x2, x_2, T+H, K), tspan, y0)
                ;
```

```
            Pert = [0; 0; 0; 0; 0];
        elseif j == 7
            sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D,
                lambda+H, x0, x1, x_1, x2, x_2, T, K), tspan, y0)
                ;
            Pert = [0; 0; 0; 0; 0];
        end
        norm_signal = deval(sol, (linspace(tspan(2)-T, tspan(2),
            nb_samples)), 1);
        coeffs_pert = MY_NORM_COEFFS(norm_signal,linspace(tspan
            (2)-T, tspan(2), nb_samples));
        Fpert = coeffs_pert - (MODEL_pred(1:5)+Pert); % added
            parenthese around the subtracted term
        Fx(1:5,j) = 1/H * (Fpert-Fref);
        %norm(deval(sol, [linspace(tspan(2)-2*T, tspan(2)-T,
            nb_samples)], 1) - norm_signal);
    end

    %% PHASE CONDITION DERIVATIVES :
    Fx(6,2) = deriv_coeffs(2);
    Fx(6,3) = deriv_coeffs(3);
    Fx(6,4) = deriv_coeffs(4);
    Fx(6,5) = deriv_coeffs(5);

    %% PSEUDO-ARC LENGTH DERIVATIVES :
    Fx(7,:) = MODEL_dv.';

end
```

The next code is the main script to launch the experiment:

```
close all
clear all
format longg

global tspan
%global y0
global nb_samples
global norm_time
global DEBUG
global options

global A
global B
global C
global D
global EPS

DEBUG = 0;
options=odeset('RelTol',1e-08,'AbsTol',1e-08);

%% SIMULATION PARAMETERS :
```

```
A = 1.8;
B = 0.0;
C = 1.0;
D = -1.0;
EPS = 0.6;
y0 = [0.8 0.8];
T = 18;                          % Period
nb_samples = 700;                % Number of samples for ODE45 time
    series.
tspan = [0 nb_samples*T];
norm_time = linspace(0, 1, nb_samples);

%% CONTINUATION PARAMETERS :
% We fix q = 2 in this code.

E = -0.17;
K = -25;                         % Gain for the control
lambda = E;                      % Continuation parameter lambda
ds = 0.005;                      % Pseudo-arclength step size
H = 0.001;                       % For the derivatives

METHOD = "NEWTON";               % USED METHOD : "NEWTON"/"BROYDEN"
VECTOR_TYPE = "SECANT";          % VECTOR COMPUTATION METHOD : "
    TANGENT"/"SECANT"
ITER = 20;                       % NEWTON/BROYDEN iteration
tol = 1e-06;                     % Tolerance for iteration method.
points = 3;                      % Number of points for continuation
    45
%% ALLOCATIONS :

% Computing a previous signal with the initial coefficients for
    the
% derivative calculation:
sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D, E, 0, 0, 0, 0,
     0, 0, 0), tspan, y0, options);
previous_signal = deval(sol, (linspace(tspan(2)-T, tspan(2),
    nb_samples)), 1);
previous_time = linspace(tspan(2)-T, tspan(2), nb_samples);
% y0=sol.y(:,end)';

coeff_ini = MY_NORM_COEFFS(previous_signal,previous_time);   %
    x0, x1, x-1
MODEL_guess = [coeff_ini; T; lambda];            % First guess :
    coefficients + T + PARAMETER
MODEL_dv = zeros(length(MODEL_guess), 1);                     %
    Direction vector initialization
MODEL_dv(length(MODEL_guess)) = 1;                            %
    Direction vector initialization : [0 0 0 0 1]

% Saving arrays for each post-iteration process:
MODEL_pred_previous = MODEL_guess;
```

```
63 MODEL_pred_stock = [MODEL_guess];
64
65 % Saving arrays for the max and the min of the good signal:
66 MAX = [max(previous_signal)];
67 MIN = [min(previous_signal)];
68
69 for p = 1:points
70     %% PREDICTION :
71     MODEL_pred = MODEL_guess+ds*MODEL_dv;
72
73     %% DERIVATIVE OF THE PREVIOUS SIGNAL :
74     deriv_coeffs = MY_DERIVATIVE_NORM_SIGNAL(previous_signal,
           previous_time);
75
76     %% ITERATION METHOD :
77     % Application of Newton or Broyden.
78     if METHOD=="NEWTON"
79         [MODEL_pred, Fx, ~] = NEWTON_METHOD_PERIOD(MODEL_pred,
               MODEL_dv, MODEL_guess, deriv_coeffs, ITER, H, ds, tol
               , K, y0);
80     elseif METHOD == "BROYDEN"
81         [MODEL_pred, Fx] = BROYDEN_METHOD_PERIOD(MODEL_pred,
               MODEL_dv, MODEL_guess, deriv_coeffs, ITER, H, ds, tol
               , K);
82     end
83
84     %% VECTOR UPDATE :
85     if VECTOR_TYPE == "TANGENT"
86         MODEL_dv = UPDATE_TANGENT(Fx);
                                                                %NEW
               GUESSES
87     elseif VECTOR_TYPE == "SECANT"
88         MODEL_dv = UPDATE_SECANT(MODEL_pred, MODEL_pred_previous
               ); % To update the secant, we just have to take the
               two last computed points: [x(p) - x(p-1); y(p) - y(p
               -1)]
89     end
90
91     %% STORAGE OF DATA:
92     MODEL_pred_previous = MODEL_pred;
93     MODEL_guess = MODEL_pred;
94     MODEL_pred_stock = [MODEL_pred_stock, MODEL_pred];
95
96     %% Extracting the max and the min of these values :
97     sol = ode45(@(t,y) FHN_ODE(t, y, EPS, A, B, C, D, MODEL_pred
           (7), MODEL_pred(1), MODEL_pred(2), MODEL_pred(3),
           MODEL_pred(4), MODEL_pred(5), MODEL_pred(6), K), tspan,
           y0);
98     previous_signal = deval(sol, (linspace(tspan(2)-MODEL_pred
           (6), tspan(2), nb_samples)), 1);
99
```

```
    MAX = [MAX, max(previous_signal)];
    MIN = [MIN, min(previous_signal)];

    disp("Point " + string(p) + " computed.")

end

figure
hold on
grid on
title("Periodic bifurcation diagram test")

scatter(MODEL_pred_stock(length(MODEL_guess),:), MAX(:), 'b','filled');
scatter(MODEL_pred_stock(length(MODEL_guess),:), MIN(:), 'g','filled');

E_stat = csvread('E_model.csv');
V_stat = csvread('V_model.csv');
XPPAUT = load("XPPAUT_FHN.dat");
plot(E_stat(2:end), V_stat(2:end), 'k--');
scatter(XPPAUT(1000:2:end, 1), XPPAUT(1000:2:end, 2), 'k');
scatter(XPPAUT(1000:2:end, 1), XPPAUT(1000:2:end, 3), 'k');
xlim([-0.25 -0.09]);
%ylim([-1.1 1.1]);
legend("Max", "Min")
hold off
```

9.5 Chapter 6: Recurrence Structure Analysis

This session will be written half in Python and Matlab. The data processing will be in Python, and the Recurrence Structure Analysis will be applied in Matlab.

9.5.1 Part 1: Reading the data, applying the K-means method, and applying the baseline alignment method.

We first propose to take a look to the importations:

```
import numpy as np
import matplotlib.pyplot as plt
import glob
import os
import scipy.io
from scipy.io import savemat
from sklearn.cluster import KMeans
import matplotlib.font_manager as font_manager
```

Then we define some functions for the preprocessing of the data:

```
def fullfill_symmetric(matrix):
    n = matrix.shape[0]
    for i in range(n):
```

```
        for j in range(i+1,n):
            matrix[i,j]=matrix[j,i]
    return matrix

def Neuron_for_each_odors(avg):

    for neuron in range(len(avg)):
        if neuron == 0 :
            Matrix = []
            for odor in range(len(avg[neuron])):
                Matrix.append([avg[neuron][odor].ravel()])
        else:
            for odor in range(len(avg[neuron])):
                Matrix[odor].append(avg[neuron][odor].ravel())

    return Matrix

def distance_matrix (data):

    matrix = np.zeros((len(data),len(data)))

    for coord1 in range(len(data)):
        for coord2 in range(len(data)):
            squared_dist = np.sum((data[coord1]-data[coord2])
                **2, axis=0)
            dist = np.sqrt(squared_dist)
            matrix[coord1][coord2] = dist

    return matrix
```

We propose first a script in order to extract the time series from the initial files:

```
print("DATA SET STUDIED ---------------------------\n")

path = "./../AndreasSchneider/CalciumImaging/"

###############################################

file_C525 = scipy.io.loadmat(path + 'C525_data.mat')

mean_C525 = file_C525['meanImagesSingle']
trial_C525 = file_C525['trialInt']
avg_C525 = file_C525['trialInt_avg']
type_C525 = scipy.io.loadmat(path + 'C525_cellTypeIdx.mat')['
    C525_cellTypeIdx']

avg_C525_Tufted = []
avg_C525_Mitral = []

trial_C525_Tufted = []
trial_C525_Mitral = []

```

```
for i in range(len(type_C525)):
    if type_C525[i][0] == 0:
        avg_C525_Tufted.append(avg_C525[i])
        trial_C525_Tufted.append(trial_C525[i])
    if type_C525[i][0] == 1:
        avg_C525_Mitral.append(avg_C525[i])
        trial_C525_Mitral.append(trial_C525[i])

avg_C525_Tufted = Neuron_for_each_odors(avg_C525_Tufted)
avg_C525_Mitral = Neuron_for_each_odors(avg_C525_Mitral)
avg_C525 = Neuron_for_each_odors(avg_C525)

trial_C525_Tufted = Neuron_for_each_odors(trial_C525_Tufted)
trial_C525_Mitral = Neuron_for_each_odors(trial_C525_Mitral)

print("C525 DATA SHAPE ---------------------------- \n\
   nmeanImagesSingle : ", np.shape(mean_C525))
print("trialInt : ", np.shape(trial_C525))
print("trialInt Tufted : ", np.shape(trial_C525_Tufted))
print("trialInt Mitral : ", np.shape(trial_C525_Mitral))
print("trialInt_avg: ", np.shape(avg_C525))
print("trialInt_avg_Tufted: ", np.shape(avg_C525_Tufted))
print("trialInt_avg_Mitral: ", np.shape(avg_C525_Mitral), "\n")

format_Tufted = {"a": avg_C525_Tufted, "label":"C525_Tufted"}
format_Mitral = {"a": avg_C525_Mitral, "label":"C525_Mitral"}

trial_Tufted = {"a": trial_C525_Tufted, "label":"C525_Tufted"}
trial_Mitral = {"a": trial_C525_Mitral, "label":"C525_Mitral"}

savemat("matlab_C525_Tufted.mat", format_Tufted)
savemat("matlab_C525_Mitral.mat", format_Mitral)

savemat("matlab_C525_trial_Tufted.mat", trial_Tufted)
savemat("matlab_C525_trial_Mitral.mat", trial_Mitral)

##############################################

file_C531 = scipy.io.loadmat(path + 'C531_data.mat')

mean_C531 = file_C531['meanImagesSingle']
trial_C531 = file_C531['trialInt']
avg_C531 = file_C531['trialInt_avg']
type_C531 = scipy.io.loadmat(path + 'C531_cellTypeIdx.mat')['
   C531_cellTypeIdx']

avg_C531_Tufted = []
avg_C531_Mitral = []

trial_C531_Tufted = []
trial_C531_Mitral = []
```

```
69
70 for i in range(len(type_C531)):
71     if type_C531[i][0] == 0:
72         avg_C531_Tufted.append(avg_C531[i])
73         trial_C531_Tufted.append(trial_C531[i])
74     if type_C531[i][0] == 1:
75         avg_C531_Mitral.append(avg_C531[i])
76         trial_C531_Mitral.append(trial_C531[i])
77
78 avg_C531_Tufted = Neuron_for_each_odors(avg_C531_Tufted)
79 avg_C531_Mitral = Neuron_for_each_odors(avg_C531_Mitral)
80 avg_C531 = Neuron_for_each_odors(avg_C531)
81
82 trial_C531_Tufted = Neuron_for_each_odors(trial_C531_Tufted)
83 trial_C531_Mitral = Neuron_for_each_odors(trial_C531_Mitral)
84
85 print("C531 DATA SHAPE _____ \n\
   nmeanImagesSingle : ", np.shape(mean_C531))
86 print("trialInt : ", np.shape(trial_C531))
87 print("trialInt Tufted : ", np.shape(trial_C531_Tufted))
88 print("trialInt Mitral : ", np.shape(trial_C531_Mitral))
89 print("trialInt_avg: ", np.shape(avg_C531))
90 print("trialInt_avg_Tufted: ", np.shape(avg_C531_Tufted))
91 print("trialInt_avg_Mitral: ", np.shape(avg_C531_Mitral), "\n")
92
93 format_Tufted = {"a": avg_C531_Tufted, "label":"C531_Tufted"}
94 format_Mitral = {"a": avg_C531_Mitral, "label":"C531_Mitral"}
95
96 trial_Tufted = {"a": trial_C531_Tufted, "label":"C531_Tufted"}
97 trial_Mitral = {"a": trial_C531_Mitral, "label":"C531_Mitral"}
98
99 savemat("matlab_C531_Tufted.mat", format_Tufted)
100 savemat("matlab_C531_Mitral.mat", format_Mitral)
101
102 savemat("matlab_C531_trial_Tufted.mat", trial_Tufted)
103 savemat("matlab_C531_trial_Mitral.mat", trial_Mitral)
104
105 ##########################################
106
107
108
109 file_C537 = scipy.io.loadmat(path + 'C537_data.mat')
110
111 mean_C537 = file_C537['meanImagesSingle']
112 trial_C537 = file_C537['trialInt']
113 avg_C537 = file_C537['trialInt_avg']
114 type_C537 = scipy.io.loadmat(path + 'C537_cellTypeIdx.mat')['
   C537_cellTypeIdx']
115
116 avg_C537_Tufted = []
117 avg_C537_Mitral = []
```

```
118
119 trial_C537_Tufted = []
120 trial_C537_Mitral = []
121
122 for i in range(len(type_C537)):
123     if type_C537[i][0] == 0:
124         avg_C537_Tufted.append(avg_C537[i])
125         trial_C537_Tufted.append(trial_C537[i])
126     if type_C537[i][0] == 1:
127         avg_C537_Mitral.append(avg_C537[i])
128         trial_C537_Mitral.append(trial_C537[i])
129
130 avg_C537_Tufted = Neuron_for_each_odors(avg_C537_Tufted)
131 avg_C537_Mitral = Neuron_for_each_odors(avg_C537_Mitral)
132 avg_C537 = Neuron_for_each_odors(avg_C537)
133
134 trial_C537_Tufted = Neuron_for_each_odors(trial_C537_Tufted)
135 trial_C537_Mitral = Neuron_for_each_odors(trial_C537_Mitral)
136
137 print("C537 DATA SHAPE ---------------------------- \n\
    nmeanImagesSingle : ", np.shape(mean_C537))
138 print("trialInt : ", np.shape(trial_C537))
139 print("trialInt Tufted : ", np.shape(trial_C537_Tufted))
140 print("trialInt Mitral : ", np.shape(trial_C537_Mitral))
141 print("trialInt_avg: ", np.shape(avg_C537))
142 print("trialInt_avg_Tufted: ", np.shape(avg_C537_Tufted))
143 print("trialInt_avg_Mitral: ", np.shape(avg_C537_Mitral), "\n")
144
145 format_Tufted = {"a": avg_C537_Tufted, "label":"C537_Tufted"}
146 format_Mitral = {"a": avg_C537_Mitral, "label":"C537_Mitral"}
147
148 trial_Tufted = {"a": trial_C537_Tufted, "label":"C537_Tufted"}
149 trial_Mitral = {"a": trial_C537_Mitral, "label":"C537_Mitral"}
150
151 savemat("matlab_C537_Tufted.mat", format_Tufted)
152 savemat("matlab_C537_Mitral.mat", format_Mitral)
153
154 savemat("matlab_C537_trial_Tufted.mat", trial_Tufted)
155 savemat("matlab_C537_trial_Mitral.mat", trial_Mitral)
156
157 ##########################################
158
159 print("\nDATA LOADED --------------------------------")
```

We propose now to extract the euclidian coordinates from the files:

```
1 Dist_matrices = []
2 Dist_tufted = []
3 Dist_mitral = []
4 name_dist = ["C525", "C531", "C537"]
5
6
```

```
data_coord_C525 = scipy.io.loadmat(path + 'C525_ROI_coordinates.mat')
Coordinates_C525 = data_coord_C525['C525_ROI_coordinates']

Coordinates_C525_Tufted = []
Coordinates_C525_Mitral = []

for i in range(len(type_C525)):
    if type_C525[i][0] == 0:
        Coordinates_C525_Tufted.append(Coordinates_C525[i])
    if type_C525[i][0] == 1:
        Coordinates_C525_Mitral.append(Coordinates_C525[i])

Coordinates_C525_Tufted = np.array(Coordinates_C525_Tufted)
Coordinates_C525_Mitral = np.array(Coordinates_C525_Mitral)

inds = np.where(np.isnan(Coordinates_C525))
Coordinates_C525[inds] = 0
matrix = distance_matrix (Coordinates_C525)
Dist_matrices.append(matrix)


inds = np.where(np.isnan(Coordinates_C525_Tufted))
Coordinates_C525_Tufted[inds] = 0
matrix = distance_matrix (Coordinates_C525_Tufted)
Dist_tufted.append(matrix)


inds = np.where(np.isnan(Coordinates_C525_Mitral))
Coordinates_C525_Mitral[inds] = max(Coordinates_C525_Mitral[:,2])
matrix = distance_matrix (Coordinates_C525_Mitral)
Dist_mitral.append(matrix)


##############################################
data_coord_C531 = scipy.io.loadmat(path + 'C531_ROI_coordinates.mat')
Coordinates_C531 = data_coord_C531['C531_ROI_coordinates']

Coordinates_C531_Tufted = []
Coordinates_C531_Mitral = []

for i in range(len(type_C531)):
    if type_C531[i][0] == 0:
        Coordinates_C531_Tufted.append(Coordinates_C531[i])
    if type_C531[i][0] == 1:
        Coordinates_C531_Mitral.append(Coordinates_C531[i])

Coordinates_C531_Tufted = np.array(Coordinates_C531_Tufted)
Coordinates_C531_Mitral = np.array(Coordinates_C531_Mitral)
```

```
55
56
57 inds = np.where(np.isnan(Coordinates_C531))
58 Coordinates_C531[inds] = 0
59 matrix = distance_matrix (Coordinates_C531)
60 Dist_matrices.append(matrix)
61
62
63 inds = np.where(np.isnan(Coordinates_C531_Tufted))
64 Coordinates_C531_Tufted[inds] = 0
65 matrix = distance_matrix (Coordinates_C531_Tufted)
66 Dist_tufted.append(matrix)
67
68
69 inds = np.where(np.isnan(Coordinates_C531_Mitral))
70 Coordinates_C531_Mitral[inds] = max(Coordinates_C525_Mitral
      [:,2])
71 matrix = distance_matrix (Coordinates_C531_Mitral)
72 Dist_mitral.append(matrix)
73
74 ##########################################
75
76 data_coord_C537 = scipy.io.loadmat(path + 'C537_ROI_coordinates.
      mat')
77 Coordinates_C537 = data_coord_C537['C537_ROI_coordinates']
78
79 Coordinates_C537_Tufted = []
80 Coordinates_C537_Mitral = []
81
82 for i in range(len(type_C537)):
83     if type_C537[i][0] == 0:
84         Coordinates_C537_Tufted.append(Coordinates_C537[i])
85     if type_C537[i][0] == 1:
86         Coordinates_C537_Mitral.append(Coordinates_C537[i])
87
88 Coordinates_C537_Tufted = np.array(Coordinates_C537_Tufted)
89 Coordinates_C537_Mitral = np.array(Coordinates_C537_Mitral)
90
91
92 inds = np.where(np.isnan(Coordinates_C537))
93 Coordinates_C537[inds] = 0
94 matrix = distance_matrix (Coordinates_C537)
95 Dist_matrices.append(matrix)
96
97 inds = np.where(np.isnan(Coordinates_C537_Tufted))
98 Coordinates_C537_Tufted[inds] = 0
99 matrix = distance_matrix (Coordinates_C537_Tufted)
100 Dist_tufted.append(matrix)
101
102
103 inds = np.where(np.isnan(Coordinates_C537_Mitral))
```

```
Coordinates_C537_Mitral[inds] = max(Coordinates_C525_Mitral
    [:,2])
matrix = distance_matrix (Coordinates_C537_Mitral)
Dist_mitral.append(matrix)
```

We propose now a code to plot all the coordinates :

```
fig = plt.figure(figsize=(20, 18), dpi=80)
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Coordinates_C525_Tufted[:,0], Coordinates_C525_Tufted
    [:,1], Coordinates_C525_Tufted[:,2], s=100, label = "Tufted
    C525")
ax.scatter(Coordinates_C531_Tufted[:,0], Coordinates_C531_Tufted
    [:,1], Coordinates_C531_Tufted[:,2], s=100, label = "Tufted
    C531")
ax.scatter(Coordinates_C537_Tufted[:,0], Coordinates_C537_Tufted
    [:,1], Coordinates_C537_Tufted[:,2], s=100, label = "Tufted
    C537")


plt.rc('text', usetex=True)

plt.locator_params(axis="x", nbins=3)
plt.locator_params(axis="y", nbins=2)
plt.locator_params(axis="z", nbins=2)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35)
ax.tick_params(axis='z', labelsize=35)
xLabel = ax.set_xlabel('$X$', linespacing=15, fontsize=45)
yLabel = ax.set_ylabel('$Y$', linespacing=5, fontsize=45)
zLabel = ax.set_zlabel('$Z$', linespacing=5, fontsize=45)
ax.dist = 10

plt.legend(prop={'size': 35})

plt.show()
```

We propose now a code apply the *K*-means:

```
# Here, I am concatenating my coordinates.
# First C525, then C531, and C537.

Coordinates_Tufted = np.concatenate((Coordinates_C525_Tufted,
    Coordinates_C531_Tufted), axis=0)
Coordinates_Tufted = np.concatenate((Coordinates_Tufted,
    Coordinates_C537_Tufted), axis=0)

X=np.column_stack((Coordinates_Tufted[:,0], Coordinates_Tufted
    [:,1]))

# I am applying here the Kmeans method. We determine first the K
    number of groups we want to form.
# At each iteration the Kmeans results are differents.
```

```
11
12 K = 10 # THIS THE PARAMETER TO CHANGE TO MODULATE THE NUMBER OF
       REGIONS OF INTEREST.
13 kmeans = KMeans(n_clusters=K, random_state=10).fit(X)
14 print("Number of sub-groups : ", max(kmeans.labels_)+1)
15
16 fig = plt.figure(figsize=(20, 18), dpi=80)
17 plt.subplot(1, 2, 1) # row 1, col 2 index 1
18 plt.scatter(Coordinates_Tufted[:,0], Coordinates_Tufted[:,2], c=
       kmeans.labels_, s=10)
19 #plt.axhline(y = Coordinates_Tufted[:,1], color="r", linestyle
       ="-")
20 plt.title("First view")
21 plt.xlabel("X-axis ")
22 plt.ylabel("Z-axis ")
23
24 plt.subplot(1, 2, 2) # index 2
25 plt.scatter(Coordinates_Tufted[:,1], Coordinates_Tufted[:,2], c=
       kmeans.labels_, s=10)
26 #plt.axhline(y=Coordinates_Tufted[:,1], color="r", linestyle
       ="-")
27 plt.title("Second view")
28 plt.xlabel("Y-axis ")
29 plt.ylabel("Z-axis ")
30
31 Coordinates_sub = np.zeros((K, 3))
32 denom = 0
33 for i in kmeans.labels_:
34     for j in range(len(kmeans.labels_)):
35         if i == kmeans.labels_[j]:
36             Coordinates_sub[i] += Coordinates_Tufted[j]
37             denom +=1
38     Coordinates_sub[i] = Coordinates_sub[i]/denom
39     denom = 0
40
41 fig = plt.figure(figsize=(20, 18), dpi=80)
42 ax = fig.add_subplot(111, projection='3d')
43 ax.scatter(Coordinates_Tufted[:,0], Coordinates_Tufted[:,1],
       Coordinates_Tufted[:,2], s=100, alpha = 0.4, c=kmeans.labels_
       )
44 ax.scatter(Coordinates_sub[:,0], Coordinates_sub[:,1],
       Coordinates_sub[:,2], color = "red", s=700, label = "Centers"
       )
45 ax.set_xlabel('X Label')
46 ax.set_ylabel('Y Label')
47 ax.set_zlabel('Z Label')
48 plt.legend()
49 plt.show()
50
51
52 fig = plt.figure(figsize=(20, 18), dpi=80)
```

```
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Coordinates_sub[:,0], Coordinates_sub[:,1], Coordinates_sub[:,2], color = "red", alpha = 0.5, s=500, label = "Centers")
for area in np.arange(0,K):
    ax.text(Coordinates_sub[area,0],Coordinates_sub[area,1], Coordinates_sub[area,2], '%s' % (str(area)), size=20, zorder=1, color='k')
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.legend()
plt.show()
```

And a good representation of the K-means results:

```
colors = ["red", "orange", "yellow", "lime", "green", "cyan", "blue", "purple", "crimson", "pink"]

fig = plt.figure(figsize=(20, 18), dpi=80)
ax = fig.add_subplot(111, projection='3d')
for i in range(len(kmeans.labels_)):
    ax.scatter(Coordinates_Tufted[i,0], Coordinates_Tufted[i,1], Coordinates_Tufted[i,2], color = colors[kmeans.labels_[i]], s=100, alpha = 0.5)
for i in range(len(Coordinates_sub)):
    ax.scatter(Coordinates_sub[i,0], Coordinates_sub[i,1], Coordinates_sub[i,2], color =  colors[i], s=700, edgecolors='k',linewidth=3, label = "Center " + str(i+1))
for area in np.arange(0,K-1):
    ax.text(Coordinates_sub[area,0]+5,Coordinates_sub[area,1]+5, Coordinates_sub[area,2]+5, '%s' % (str(area+1)), size=50, zorder=10000, color='k')
ax.text(Coordinates_sub[K-1,0]+5,Coordinates_sub[K-1,1]+5, Coordinates_sub[K-1,2]+10, '%s' % (str(K-1+1)), size=50, zorder=10000, color='k')
plt.rc('text', usetex=True)

plt.locator_params(axis="x", nbins=3)
plt.locator_params(axis="y", nbins=2)
plt.locator_params(axis="z", nbins=2)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35)
ax.tick_params(axis='z', labelsize=35)
xLabel = ax.set_xlabel('$X$', linespacing=15, fontsize=45)
yLabel = ax.set_ylabel('$Y$', linespacing=5, fontsize=45)
zLabel = ax.set_zlabel('$Z$', linespacing=5, fontsize=45)
ax.dist = 10


plt.show()
```

Then we compute the region of interest new time series based on the mean of all their neurons:

```
ODORS = 48

full_C525_activity = []
full_C531_activity = []
full_C537_activity = []

for STUDIED_ODOR in range(ODORS):
    C525_activity = []
    C531_activity = []
    C537_activity = []

    ########## C525 :

    end = len(avg_C525_Tufted[STUDIED_ODOR])
    C525_label = kmeans.labels_[:end]

    denom = 0
    for i in np.arange(0,K):
        ts = np.zeros(len(avg_C525_Tufted[STUDIED_ODOR][0]))
        for j in range(len(avg_C525_Tufted[STUDIED_ODOR])):
            if i == C525_label[j]:
                ts = ts + avg_C525_Tufted[STUDIED_ODOR][j]
                denom +=1
        C525_activity.append(ts/denom)
        denom = 0

    ########## C531 :

    start = end
    end = start + len(avg_C531_Tufted[STUDIED_ODOR])
    C531_label = kmeans.labels_[start:end]

    denom = 0
    for i in np.arange(0,K):
        ts = np.zeros(len(avg_C531_Tufted[STUDIED_ODOR][0]))
        for j in range(len(avg_C531_Tufted[STUDIED_ODOR])):
            if i == C531_label[j]:
                ts = ts + avg_C531_Tufted[STUDIED_ODOR][j]
                denom +=1
        C531_activity.append(ts/denom)
        denom = 0

    ########## C537 :

    start = end
    end = start + len(avg_C537_Tufted[STUDIED_ODOR])
    C537_label = kmeans.labels_[start:end]

    denom = 0
    for i in np.arange(0,K):
        ts = np.zeros(len(avg_C537_Tufted[STUDIED_ODOR][0]))
```

```
        for j in range(len(avg_C537_Tufted[STUDIED_ODOR])):
            if i == C537_label[j]:
                ts = ts + avg_C531_Tufted[STUDIED_ODOR][j]
                denom +=1
        C537_activity.append(ts/denom)
        denom = 0

    full_C525_activity.append(C525_activity)
    full_C531_activity.append(C531_activity)
    full_C537_activity.append(C537_activity)
```

And so the baseline alignment method with some visualisations :

```
# Removing the NaN values.

for odor in range(len(full_C525_activity)):
    for ts in range(len(full_C525_activity[odor])):
        full_C525_activity[odor][ts][np.isnan(full_C525_activity
            [odor][ts])] = 0

for odor in range(len(full_C531_activity)):
    for ts in range(len(full_C531_activity[odor])):
        full_C531_activity[odor][ts][np.isnan(full_C531_activity
            [odor][ts])] = 0

for odor in range(len(full_C537_activity)):
    for ts in range(len(full_C537_activity[odor])):
        full_C537_activity[odor][ts][np.isnan(full_C537_activity
            [odor][ts])] = 0

# Applying the Baseline Alignment on the subject C525 :

Pre_time = 8
Zeta = np.copy(full_C525_activity)

for odor in range(len(Zeta)):
    for ts in range(len(Zeta[odor])):
        Bi = np.mean(Zeta[odor][ts][:Pre_time])
        Zeta[odor][ts] = Zeta[odor][ts] - Bi
    Mean_through_N = np.mean(Zeta[odor], axis = 0)
    for ts in range(len(Zeta[odor])):
        Zeta[odor][ts] = Zeta[odor][ts] + Mean_through_N

full_C525_activity = np.copy(Zeta)

# Applying the Baseline Alignment on the subject C531 :

Zeta = np.copy(full_C531_activity)

for odor in range(len(Zeta)):
    for ts in range(len(Zeta[odor])):
        Bi = np.mean(Zeta[odor][ts][:Pre_time])
```

```
            Zeta[odor][ts] = Zeta[odor][ts] - Bi
        Mean_through_N = np.mean(Zeta[odor], axis = 0)
        for ts in range(len(Zeta[odor])):
            Zeta[odor][ts] = Zeta[odor][ts] + Mean_through_N

full_C531_activity = np.copy(Zeta)

# Applying the Baseline Alignment on the subject C537 :

Zeta = np.copy(full_C537_activity)

for odor in range(len(Zeta)):
    for ts in range(len(Zeta[odor])):
        Bi = np.mean(Zeta[odor][ts][:Pre_time])
        Zeta[odor][ts] = Zeta[odor][ts] - Bi
    Mean_through_N = np.mean(Zeta[odor], axis = 0)
    for ts in range(len(Zeta[odor])):
        Zeta[odor][ts] = Zeta[odor][ts] + Mean_through_N

full_C537_activity = np.copy(Zeta)

ODOR = 0

time = (np.arange(0,len(full_C525_activity[ODOR][0]))/5)*2

fig_dims = (20, 10)
fig, ax = plt.subplots(figsize=fig_dims)
for i in np.arange(0,K):
    plt.plot(time, np.transpose(full_C525_activity[ODOR][i]),
       label = "Subgroup " + str(i+1))

ax.axvspan((8/5)*2, (13/5)*2, alpha=0.2, color='red')

plt.rc('text', usetex=True)

plt.locator_params(axis="x", nbins=3)
plt.locator_params(axis="y", nbins=2)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35)
plt.xlabel("Time (s)", linespacing=5, fontsize=35)
plt.ylabel("Calcium activity", linespacing=5, fontsize=35)
plt.title("C525 after process", fontsize=40)
plt.legend(fontsize=20)
plt.show()

fig_dims = (20, 10)
fig, ax = plt.subplots(figsize=fig_dims)
for i in np.arange(0,K):
    plt.plot(time, np.transpose(full_C531_activity[ODOR][i]),
       label = "Subgroup " + str(i+1))

```

```
ax.axvspan((8/5)*2, (13/5)*2, alpha=0.2, color='red')

plt.locator_params(axis="x", nbins=3)
plt.locator_params(axis="y", nbins=2)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35)
plt.xlabel("Time (s)", linespacing=5, fontsize=35)
plt.ylabel("Calcium activity", linespacing=5, fontsize=35)
plt.title("C531 after process", fontsize=40)
plt.legend(loc = 1,fontsize=20)
plt.show()

fig_dims = (20, 10)
fig, ax = plt.subplots(figsize=fig_dims)
for i in np.arange(0,K):
    plt.plot(time, np.transpose(full_C537_activity[0][i]),
       label = "Subgroup " + str(i+1))

ax.axvspan((8/5)*2, (13/5)*2, alpha=0.2, color='red')

plt.locator_params(axis="x", nbins=3)
plt.locator_params(axis="y", nbins=2)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35)
plt.xlabel("Time (s)", linespacing=5, fontsize=35)
plt.ylabel("Calcium activity", linespacing=5, fontsize=35)
plt.title("C537 after process", fontsize=40)
plt.legend(fontsize=20)
plt.show()
```

At the end, we record the data into matlab files for the RSA code :

```
format_C525 = {"a": full_C525_activity, "label":"C525_Tufted"}
savemat("./RSA/matlab_C525_peter_10.mat", format_C525)

format_C531 = {"a": full_C531_activity, "label":"C531_Tufted"}
savemat("./RSA/matlab_C531_peter_10.mat", format_C531)

format_C537 = {"a": full_C537_activity, "label":"C537_Tufted"}
savemat("./RSA/matlab_C537_peter_10.mat", format_C537)
```

9.5.2 Part 2: Creating the centroid heatmaps for one odor and for all subjects.

For this part, the importations are:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import glob
import os
import scipy.io
import matplotlib.font_manager as font_manager
from matplotlib import cm as cm
```

```
from mpl_toolkits.axes_grid1 import AxesGrid

from scipy.io import savemat
from skimage import measure
from sklearn.cluster import KMeans
from scipy.ndimage.filters import gaussian_filter

from sklearn import preprocessing
```

After applying the RSA, we extract the centroids, we keep the same variables than in the previous subsection and we represent the metastable centroid heatmap for one subject:

```
#### CHOOSE THE ODOR AND SUBJECT :
df = pd.read_csv('../Centroids/Subject_1_Ester_28_centroid.csv',
    header = None)
df = df.to_numpy()

s = 60 # Parameter for the Gaussian filter
bins=1000 # Parameter for the heatmap representation

fig = plt.figure(figsize=(35, 12), dpi=80)
ax = fig.gca()

LABEL = np.arange(1,len(df[0])+1).astype(str)
for i in range(len(LABEL)):
    LABEL[i] = "Metastable state " + LABEL[i]
plt.plot(np.arange(1,11), df, label = LABEL)
plt.plot(np.arange(1,11), np.zeros(len(np.arange(1,11))), 'k--',
    alpha = 0.5)

plt.rc('text', usetex=True)
plt.xlim([1,10])
plt.locator_params(axis="x", nbins=10)
plt.locator_params(axis="y", nbins=3)
ax.tick_params(axis='x', labelsize=45)
ax.tick_params(axis='y', labelsize=45)
xLabel = ax.set_xlabel('K group', linespacing=3, fontsize=50)
yLabel = ax.set_ylabel('Center of recurrence value', rotation
   =90, linespacing=3, fontsize=56)
ax.dist = 10
plt.legend(prop={'size': 35})
plt.show()

##############################################################

s = 60

if np.shape(df)[1] == 1 :
    fig = plt.figure(figsize=(8, 6), dpi=80) # 2
if np.shape(df)[1] == 2 :
    fig = plt.figure(figsize=(10, 9), dpi=80) # 2
if np.shape(df)[1] == 3 :
```

```
    fig = plt.figure(figsize=(10, 9), dpi=80) # 2
if np.shape(df)[1] == 4 :
    fig = plt.figure(figsize=(25, 23), dpi=80) # 4

for state in range(len(df[0])):


    #### CHOOSE THE METASTABLE STATE :
    centroid = df[:,state]

    x = list(map(int, Coordinates_Tufted[:,0]))
    y = list(map(int, Coordinates_Tufted[:,1]))
    z = np.zeros(len(x))

    heatmap, xedges, yedges = np.histogram2d(x, y, bins=bins)

    for i in range(len(z)):
        z[i] = centroid[kmeans.labels_[i]]

    x_center = x - np.min(x)
    y_center = y - np.min(y)

    Z = np.zeros((np.max(x)-np.min(x)+30, np.max(y)-np.min(y)
       +30))

    for i in range(len(x_center)):
        Z[x_center[i]][y_center[i]] = z[i]


        ##############################################

    ax = plt.subplot(1, len(df[0]), state+1)

    heatmap = Z
    heatmap = gaussian_filter(heatmap, sigma=s).T
    extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
    plt.imshow(heatmap, origin='lower', cmap=cm.jet)
    cbar = plt.colorbar(fraction=0.046, pad=0.04, orientation='
       horizontal')
    if np.shape(df)[1] == 1 :
        plt.title("Metastable state " + str(int(state+1)),
           fontsize=25)
        cbar.ax.tick_params(labelsize=20)
    if np.shape(df)[1] == 2 :
        plt.title("Metastable state " + str(int(state+1)),
           fontsize=40)
        cbar.ax.tick_params(labelsize=20)
    if np.shape(df)[1] == 3 :
        plt.title("Metastable state " + str(int(state+1)),
           fontsize=25)
        cbar.ax.tick_params(labelsize=15)
```

```
    if np.shape(df)[1] == 4 :
        plt.title("Metastable state " + str(int(state+1)),
            fontsize=35)
        cbar.ax.tick_params(labelsize=30)
    #plt.clim(-0.0002, 0.15)

    #for i in range(len(x)):
    #    plt.scatter(x[i], y[i], color = colors[kmeans.labels_[i
        ]], s=100)

    for i in range(len(Coordinates_sub)):
        plt.scatter(Coordinates_sub[i,0], Coordinates_sub[i,1],
            color =  colors[i], s=400, edgecolors='k',linewidth
            =3, label = "Center " + str(i+1))
    #for area in np.arange(0,K):
    #    plt.text(Coordinates_sub[area,0]+15,Coordinates_sub[
        area,1]+35,  '%s' % (str(area+1)), size=35, zorder=10000,
          color='k')
    ax.tick_params(axis='x', labelsize=25)
    ax.tick_params(axis='y', labelsize=25)
    plt.xticks([])
    plt.yticks([])

fig.tight_layout()
# Put a legend to the right of the current axis
ax.legend(loc='center left', bbox_to_anchor=(1., 0.5), prop={'
    size': 25})
plt.show()
```

9.5.3 Part 3: Creating the centroid heatmaps for reach odor type.

For this part, the importations are:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import glob
import os
import scipy.io
from matplotlib import cm as cm
from mpl_toolkits.axes_grid1 import AxesGrid
import matplotlib.font_manager as font_manager
from matplotlib import cm as cm
from mpl_toolkits.axes_grid1 import AxesGrid

from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import SpectralClustering
from scipy.ndimage.filters import gaussian_filter
```

```

import seaborn as sns
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics
from scipy.stats import bootstrap

#Import resampling and modeling algorithms
from sklearn.utils import resample # for Bootstrap sampling
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors
from sklearn.decomposition import PCA

#KFold CV
from sklearn.model_selection import KFold, LeaveOneOut
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import LabelEncoder

import tpot
from tpot import TPOTClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from collections import Counter
from imblearn.over_sampling import SMOTE

import warnings

warnings.filterwarnings("ignore")

MODE = ['accuracy','precision_micro', 'recall_micro']
colors = ["red", "orange", "yellow", "lime", "green", "cyan", "
    blue", "purple", "black", "pink"]
```

After applying the RSA, we extract the centroids again, we keep the same variables than in 9.5.1 and we represent the proportions of each category:

```
path = "./../Centroids/"
motif = "Subject*.csv"

LABEL = []

for fichier in glob.iglob(os.path.join(path, motif)):
    if os.path.isfile(fichier):
        string = fichier[25:]
```

```
        newstring = ''.join([i for i in string if not i.isdigit
            ()])
        name = newstring[:-14]

        df = pd.read_csv(fichier, header = None)
        df = df.to_numpy()

        for i in range(np.shape(df)[1]):
            LABEL.append(name)

LABEL = np.asarray(LABEL)
print("NUMBER OF NAME : ", np.shape(LABEL))

LABEL_NUM = np.zeros(len(LABEL))

A = 0
B = 0
C = 0
D = 0
E = 0
F = 0
G = 0

for i in range(len(LABEL)):
    if LABEL[i] == "Acid":
        A += 1
    if LABEL[i] == "Alcohol":
        B += 1
    if LABEL[i] == "Aldehyde":
        C += 1
    if LABEL[i] == "Aromatic":
        D += 1
    if LABEL[i] == "Ester":
        E += 1
    if LABEL[i] == "Ketone":
        F  += 1
    if LABEL[i] == "Control":
        G  += 1

print("PROPOTIONS")
print("Acid : ", A/len(LABEL)*100, " %")
print("Alcohol : ", B/len(LABEL)*100, " %")
print("Aldehyde : ", C/len(LABEL)*100, " %")
print("Aromatic : ", D/len(LABEL)*100, " %")
print("Ester : ", E/len(LABEL)*100, " %")
print("Ketone : ", F/len(LABEL)*100, " %")
print("Control : ", G/len(LABEL)*100, " %")

sizes = np.array([A,B,C,D,E,F,G])

print("\nIMBALANCED RATIO : ", np.max(sizes)/np.min(sizes))
```

```
59 print("With this definition, Imbalance ratio >= 1, the value 1
       corresponding to perfectly balanced data. In our example, it
       is 99.9/0.1 = 999.")
60
61
62 print("\nBOOTSTRAP REPRESENTATION AFTER RESAMPLING OF 500: ",
       500*100/308, " %")
63
64 fig1, ax1 = plt.subplots(figsize=(20, 18), dpi=80)
65 plt.rcParams.update({'font.size': 22})
66 ax1.pie(sizes, labels=["ACIDS", "ALCOHOLS", "ALDEHYDES", "
       AROMATICS", "KETONES", "ESTERS", "CONTROLS"], autopct='%1.1f
       %%',
67         shadow=False, startangle=20)
68 ax1.axis('equal')  # Equal aspect ratio ensures that pie is
       drawn as a circle.
69 plt.rc('text', usetex=True)
70 ax1.tick_params(axis='x', labelsize=35)
71 ax1.tick_params(axis='y', labelsize=35, rotation = 1)
72 #plt.legend(prop={'size': 20})
73 plt.show()
```

Then, we apply the SMOTE methodology on our new data set:

```
1  path = "./../Centroids/"
2  motif = "Subject*.csv"
3
4  DATA = []
5  LABEL = []
6
7  for fichier in glob.iglob(os.path.join(path, motif)):
8      if os.path.isfile(fichier):
9          string = fichier[25:]
10         newstring = ''.join([i for i in string if not i.isdigit
               ()])
11         name = newstring[:-14]
12
13         df = pd.read_csv(fichier, header = None)
14         df = df.to_numpy()
15
16         for i in range(np.shape(df)[1]):
17             DATA.append(df[:,i])
18             LABEL.append(name)
19
20 DATA = np.asarray(DATA)
21 LABEL = np.asarray(LABEL)
22 print("DATA SHAPE : ", np.shape(DATA))
23 print("NUMBER OF NAME : ", np.shape(LABEL))
24
25 LABEL = LabelEncoder().fit_transform(LABEL)
26 # summarize distribution
27 counter = Counter(LABEL)
```

```
for k,v in counter.items():
    per = v / len(LABEL) * 100
    print('Class=%d, n=%d (%.3f%%)' % (k, v, per))
# plot the distribution
plt.bar(counter.keys(), counter.values())
plt.show()

# label encode the target variable
LABEL = LabelEncoder().fit_transform(LABEL)
# transform the dataset
oversample = SMOTE()
DATA, LABEL = oversample.fit_resample(DATA, LABEL)
# summarize distribution
counter = Counter(LABEL)
for k,v in counter.items():
     per = v / len(LABEL) * 100
     print('Class=%d, n=%d (%.3f%%)' % (k, v, per))
# plot the distribution
plt.bar(counter.keys(), counter.values())
plt.show()
```

And we apply the KNN classifier on the resampled data-set with the confusion matrices:

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(DATA, LABEL,
    test_size=0.25) # 70% training and 30% test

#Create a svm Classifier
clf = KNeighborsClassifier(n_neighbors=4, metric='cosine')

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision Score : ",metrics.precision_score(y_test,y_pred
   ,pos_label='positive',average='micro'))
print("Recall Score :" , metrics.recall_score(y_test, y_pred,
   pos_label='positive',average='micro') )

CONF = metrics.confusion_matrix(y_test, y_pred)
NEW_CONF = np.zeros((len(CONF), len(CONF)))

for i in range(len(CONF)):
    dfmin = np.min(CONF[i])
    dfmax = np.max(CONF[i])
    for j in range(len(CONF[i])):
        NEW_CONF[i][j] = (CONF[i][j] - dfmin)/(dfmax - dfmin)
```

```

XNAME = ["ACI", "ALC", "ALD", "ARO", "KET", "EST", "CON"]
YNAME = ["ACI", "ALC", "ALD", "ARO", "KET", "EST", "CON"]

fig= plt.subplots(figsize=(20, 18), dpi=80)
plt.rcParams.update({'font.size': 22})
ax = sns.heatmap(NEW_CONF, xticklabels = XNAME, yticklabels =
    YNAME)


ax.collections[0].colorbar.set_label('Normalized representation'
    , labelpad=20, fontsize=14)

ax.figure.axes[-1].yaxis.label.set_size(70)


# use matplotlib.colorbar.Colorbar object
cbar = ax.collections[0].colorbar
# here set the labelsize by 20
cbar.ax.tick_params(labelsize=50)

plt.rc('text', usetex=True)
ax.tick_params(axis='x', labelsize=35)
ax.tick_params(axis='y', labelsize=35, rotation = 1)
xLabel = ax.set_xlabel('True label', linespacing=10, fontsize
    =70)
yLabel = ax.set_ylabel('Predicted label',  rotation=90,
    linespacing=10, fontsize=70)
ax.dist = 10
plt.legend(prop={'size': 20})
plt.show()
```

Now, we propose a code to compute the heatmaps based on the good classifications for each odor category (acid, alcohol, aldehyde, aromtic, ketone, ester, control) :

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(DATA, LABEL,
     test_size=0.25) # 70% training and 30% test

#Create a svm Classifier
clf = KNeighborsClassifier(n_neighbors=5, metric='cosine')

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

A = np.zeros(10)
B = np.zeros(10)
C = np.zeros(10)
```

```
D = np.zeros(10)
E = np.zeros(10)
F = np.zeros(10)
G = np.zeros(10)

Ac = 0
Bc = 0
Cc = 0
Dc = 0
Ec = 0
Fc = 0
Gc = 0

for i in range(len(y_pred)):
    if y_pred[i] == y_test[i]:
        if y_pred[i] == 0:
            A += np.array(X_test)[i]
            Ac += 1
        if y_pred[i] == 1:
            B += np.array(X_test)[i]
            Bc += 1
        if y_pred[i] == 2:
            C += np.array(X_test)[i]
            Cc += 1
        if y_pred[i] == 3:
            D += np.array(X_test)[i]
            Dc += 1
        if y_pred[i] == 4:
            E += np.array(X_test)[i]
            Ec += 1
        if y_pred[i] == 5:
            F += np.array(X_test)[i]
            Fc += 1
        if y_pred[i] == 6:
            G += np.array(X_test)[i]
            Gc += 1

A = A/Ac
B = B/Bc
C = C/Cc
D = D/Dc
E = E/Ec
F = F/Fc
G = G/Gc

df = np.array([A, B, C, D, E, F, G])
LABEL = np.array(["ACIDS", "ALCOHOLS", "ALDEHYDES", "AROMATICS",
    "KETONES", "ESTERS", "CONTROLS"])

s = 60
bins=1000
```

```
66
67  # Generate some test data
68  x = Coordinates[:,0]
69  y = Coordinates[:,1]
70  z = np.zeros(len(x))
71
72  heatmap, xedges, yedges = np.histogram2d(x, y, bins=bins)
73
74  maxi = np.max(df)
75  mini = np.min(df)
76
77  df_norm = np.copy(df)
78  df_norm = (df_norm -np.min(df_norm ))/(np.max(df_norm )-np.min(
        df_norm ))
79
80  fig = plt.figure(figsize=(35, 12), dpi=80)
81  ax = fig.gca()
82
83  for i in range(len(df_norm)):
84      plt.plot(np.arange(1,11), df_norm[i], linewidth = 5, label =
            LABEL[i])
85  plt.plot(np.arange(1,11), np.zeros(len(np.arange(1,11))), 'k--',
        alpha = 0.5)
86
87  plt.rc('text', usetex=True)
88  plt.xlim([1,10])
89  plt.ylim([-0.1,1.1])
90  plt.locator_params(axis="x", nbins=10)
91  plt.locator_params(axis="y", nbins=3)
92  ax.tick_params(axis='x', labelsize=45)
93  ax.tick_params(axis='y', labelsize=45)
94  xLabel = ax.set_xlabel('K group', linespacing=3, fontsize=50)
95  yLabel = ax.set_ylabel('Center of recurrence value', rotation
        =90, linespacing=3, fontsize=56)
96  ax.dist = 10
97  #plt.legend(prop={'size': 35})
98  plt.show()
99
100 ##############################################################
101
102 s = 60
103
104
105 for state in range(len(df_norm)):
106
107     fig = plt.figure(figsize=(75, 75), dpi=80) # 4
108
109     #### CHOOSE THE METASTABLE STATE :
110     centroid = df_norm[state]
111
112     x = list(map(int, Coordinates[:,0]))
```

```
113     y = list(map(int, Coordinates[:,1]))
114     z = np.zeros(len(x))
115 
116     for i in range(len(z)):
117         z[i] = centroid[kmeans.labels_[i]]
118 
119     x_center = x - np.min(x)
120     y_center = y - np.min(y)
121 
122     Z = np.zeros((np.max(x)-np.min(x)+30, np.max(y)-np.min(y)
        +30))
123 
124     for i in range(len(x_center)):
125         Z[x_center[i]][y_center[i]] = z[i]
126         Z[x_center[i-1]][y_center[i]] = z[i]
127         Z[x_center[i]][y_center[i-1]] = z[i]
128         Z[x_center[i-1]][y_center[i-1]] = z[i]
129 
130 
131 
132     ############################################################
133 
134     ax = plt.subplot(1, len(df_norm[0]), state+1)
135 
136     heatmap = Z
137     heatmap = (heatmap -np.min(heatmap))/(np.max(heatmap)-np.min
        (heatmap))
138     #print(heatmap[np.nonzero(heatmap)])
139     heatmap = gaussian_filter(heatmap, sigma=s).T
140     extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
141     plt.imshow(heatmap, origin='lower', cmap=cm.jet,
        interpolation="gaussian")
142 
143     cbar = plt.colorbar(fraction=0.046, pad=0.004, orientation='
        horizontal')
144     if np.shape(df_norm)[1] == 1 :
145         plt.title("Metastable state " + str(int(state+1)),
            fontsize=25)
146         cbar.ax.tick_params(labelsize=20)
147     if np.shape(df_norm)[1] == 2 :
148         plt.title("Metastable state " + str(int(state+1)),
            fontsize=40)
149         cbar.ax.tick_params(labelsize=20)
150     if np.shape(df_norm)[1] == 3 :
151         plt.title("Metastable state " + str(int(state+1)),
            fontsize=25)
152         cbar.ax.tick_params(labelsize=15)
153     if np.shape(df_norm)[1] == 4 :
154         plt.title("Metastable state " + str(int(state+1)),
            fontsize=35)
155         cbar.ax.tick_params(labelsize=30)
```

```
    plt.clim(0, 0.025)

    #for i in range(len(x)):
    #    plt.scatter(x[i], y[i], color = colors[kmeans.labels_[i
        ]], s=100)

    for i in range(len(Coordinates_sub)):
        plt.scatter(Coordinates_sub[i,0], Coordinates_sub[i,1],
            color = colors[i], s=400, edgecolors='k',linewidth
            =3, label = "Center " + str(i+1))
    #for area in np.arange(0,K):
    #    plt.text(Coordinates_sub[area,0]+15,Coordinates_sub[
        area,1]+35, '%s' % (str(area+1)), size=35, zorder=10000,
            color='k')
    ax.tick_params(axis='x', labelsize=25)
    ax.tick_params(axis='y', labelsize=25)
    plt.xticks([])
    plt.yticks([])
    plt.title(LABEL[state])
    ax.set_xlim(0,474)
    ax.set_ylim(0,454)

    fig.tight_layout()
    # Put a legend to the right of the current axis
    ax.legend(loc='center left', bbox_to_anchor=(1., 0.5), prop
        ={'size': 25})
    plt.show()
```

For the bootstrap method, we use simply this method:

```
m = len(DATA)
nsample = 500
X_NEW_DATA = []
Y_NEW_DATA = []

for n in range(nsample):
    idx = np.random.randint(0,m)
    X_NEW_DATA.append(DATA[idx])
    Y_NEW_DATA.append(LABEL_NUM[idx])

X_NEW_DATA = np.asarray(X_NEW_DATA)
Y_NEW_DATA = np.asarray(Y_NEW_DATA)
```

Bibliography

[1] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". *The Journal of Physiology* 117.4 (1952), pp. 500–544.

[2] A. L. Hodgkin and B. Katz. "The effect of sodium ions on the electrical activity of the giant axon of the squid". *The Journal of Physiology* 108 (1949), pp. 37–77.

[3] A. L. Hodgkin, A. F. Huxley, and B. Katz. "Measurement of current-voltage relations in the membrane of the giant axon of Loligo". *The Journal of Physiology* 116.4 (1952), pp. 424–448.

[4] R. FitzHugh. "Impulses and Physiological States in Theoretical Models of Nerve Membrane". *Biophysical journal* 1.6 (1961), pp. 445–66.

[5] J. Nagumo, S. Arimoto, and S. Yoshizawa. "An Active Pulse Transmission Line Simulating Nerve Axon". *Proceedings of the IRE* 50.10 (1962), pp. 2061–2070.

[6] C. Morris and H. Lecar. "Voltage Oscillations in the Barnacle Giant Muscle Fiber". *Biophysical Journal* 35.1 (1981), pp. 193–213.

[7] G. Girier, M. Desroches, and S. Rodrigues. "From integrator to resonator neurons: A multiple-timescale scenario". *Nonlinear Dynamics* 111 (2023), pp. 16545–16556.

[8] C. Schmidt-Hieber, P. Jonas, and J. Bischofberger. "Enhanced synaptic plasticity in newly generated granule cells of the adult hippocampus". *Nature* 429.6988 (2004), pp. 184–187.

[9] S. Ge et al. "GABA sets the tempo for activity-dependent adult neurogenesis". *Trends in Neurosciences* 30.1 (2007), pp. 1–8.

[10] J. B. Aimone, W. Deng, and F. H. Gage. "Resolving new memories: a critical look at the dentate gyrus, adult neurogenesis, and pattern separation". *Neuron* 70.4 (2011), pp. 589–596.

[11] A. Marín-Burgin et al. "Unique processing during a period of high excitation/inhibition balance in adult-born neurons". *Science* 335.6073 (2012), pp. 1238–1242.

[12] C. V. Dieni et al. "Distinct Determinants of Sparse Activation during Granule Cell Maturation". *Journal of Neuroscience* 33.49 (2013), pp. 19131–19142.

[13] J. Brunner et al. "Adult-born granule cells mature through two functionally distinct states". *Elife* 3 (2014), e03104.

[14] C. V. Dieni et al. "Low excitatory innervation balances high intrinsic excitability of immature dentate neurons". *Nature Communications* 7.1 (2016), pp. 1–13.

[15] J. Danielewicz et al. "Complex excitability and "flipping" of granule cells: an experimental and computational study". HAL e-print hal-04232000. Oct. 2023.

[16] Y. A. Kuznetsov. *Elements of applied bifurcation theory*. 2nd ed. Applied Mathematical Sciences 112. Berlin: Springer, 1998.

[17] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction*. Springer Verlag, 1990.

[18] E. J. Doedel. “AUTO: A program for the automatic bifurcation analysis of autonomous systems”. *Congressus Numerantium* 30 (1981), pp. 265–284.

[19] D. Amakhin et al. “Observing hidden neuronal states in experiments”. arXiv e-print 2308.15477. Aug. 2023.

[20] H. Poincaré. “Sur le problème des trois corps et les équations de la dynamique”. *Acta Math.* 13.1 (1890), A3–A270.

[21] J.-P. Eckmann, S. Kamphorst, and D. Ruelle. “Recurrence Plots of Dynamical Systems”. *Europhysics Letters* 4.9 (1987), pp. 973–977.

[22] P. beim Graben and A. Hutt. “Detecting recurrence domains of dynamical systems by symbolic dynamics”. *Physical Review Letters* 110.15 (15 2013), p. 154101.

[23] P. beim Graben and A. Hutt. “Detecting event-related recurrences by symbolic analysis: Applications to human language processing”. *Philosophical Transactions of the Royal Society A* 373.2034 (2015), p. 20140089.

[24] P. beim Graben et al. “Metastable resting state brain dynamics”. *Frontiers in Computational Neuroscience* 13 (2019), p. 62.

[25] T. Tošić et al. “Statistical frequency-dependent analysis of trial-to-trial variability in single time series by recurrence plots”. *Frontiers in Systems Neuroscience* 9.184 (2016), p. 184.

[26] M. Desroches et al. “Mixed-Mode Oscillations with Multiple Time Scales”. *SIAM Review* 54.2 (2012), pp. 211–288.

[27] M. Desroches, M. Krupa, and S. Rodrigues. “Spike-adding in parabolic bursters: The role of folded-saddle canards”. *Physica D: Nonlinear Phenomena* 331 (2016), pp. 58–70.

[28] M. Desroches et al. “Canards, Folded Nodes, and Mixed-Mode Oscillations in Piecewise-Linear Slow-Fast Systems”. *SIAM Review* 58.4 (2016), pp. 653–691.

[29] E. M. Izhikevich. “Neural excitability, spiking and bursting”. *International Journal of Bifurcation and Chaos* 10.06 (2000), pp. 1171–1266.

[30] E. M. Izhikevich. “Resonate-and-fire neurons”. *Neural Networks* 14.6-7 (2001), pp. 883–894.

[31] S. Prescott et al. “Pyramidal Neurons Switch From Integrators In Vitro To Resonators Under In Vivo-Like Conditions”. *Journal of Neurophysiology* 100.6 (2008), pp. 3030–3042.

[32] L. F. Abbott. “Lapicque’s Introduction of the Integrate-and-Fire Model Neuron (1907)”. *Brain Research Bulletin* 50.5-6 (1999), pp. 303–304.

[33] A. L. Hodgkin. “The Local Electric Changes Associated with Repetitive Action in a Non-Myelinated Axon”. *The Journal of Physiology* 107.2 (1948), pp. 165–181.

[34] E. Izhikevich. *Dynamical Systems in Neuroscience*. Computational neuroscience Dynamical systems in neuroscience. MIT Press, 2007.

[35] E. J. Doedel. *In: Numerical Continuation Methods for Dynamical Systems*. Ed. by B. Krauskopf, H. M. Osinga, and J. G. Vioque. New York: Springer, 2007. Chap. 1, pp. 1–49.

[36] J.-P. Dedieu. “Newton-Raphson Method”. *Encyclopedia of Applied and Computational Mathematics*. Ed. by B. Engquist. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1023–1028.

[37] H. Keller, A. Nandakumaran, and M. Ramaswamy. “Lectures on Numerical Methods in Bifurcation Problems”. Lectures on Mathematics and Physics. Tata Institute of Fundamental Research, 1987.

[38] E. Doedel, H. B. Keller, and J. P. Kernevez. "Numerical Analysis and Control of Bifurcation Problems (I): Bifurcation in Finite Dimensions". *International Journal of Bifurcation and Chaos* 01.03 (1991), pp. 493–520.

[39] E. Doedel, H. B. Keller, and J. P. Kernevez. "Numerical Analysis and Control of Bifurcation Problems (II): Bifurcation in Infinite Dimensions". *International Journal of Bifurcation and Chaos* 01.04 (1991), pp. 745–772.

[40] E. Doedel. *An Introduction to Numerical Continuation Methods with Applications*. IIMAS-UNAM, 2014.

[41] C. Kirst et al. "Fundamental structure and modulation of neuronal excitability: synaptic control of coding, resonance, and network synchronization". bioRxiv e-print 10.1101/022475. July 2015.

[42] J.-M. Fellous et al. "Frequency Dependence of Spike Timing Reliability in Cortical Pyramidal Cells and Interneurons". *Journal of Neurophysiology* 85.4 (2001), pp. 1782–7.

[43] G. Yi et al. "Exploring how extracellular electric field modulates neuron activity through dynamical analysis of a two-compartment neuron model". *Journal of Computational Neuroscience* 36 (2013), pp. 383–399.

[44] J. Feldmann et al. "All-optical spiking neurosynaptic networks with self-learning capabilities". *Nature* 569.7755 (2019), pp. 208–214.

[45] J. Roach et al. "Acetylcholine Mediates Dynamic Switching Between Information Coding Schemes in Neuronal Networks". *Frontiers in Systems Neuroscience* 13 (2019), p. 64.

[46] I. Al-Darabsah and S. Campbell. "M-current induced Bogdanov–Takens bifurcation and switching of neuron excitability class". *The Journal of Mathematical Neuroscience* 11 (2021), p. 5.

[47] R. Guantes and G. Polavieja. "Variability in noise-driven integrator neurons". *Physical Review E* 71.1 (2005), p. 011911.

[48] C. Kirst, A. Herz, and M. Stemmler. "From Integrator to Resonator: The Effect of Dendritic Geometry on Neuronal Excitability". *Frontiers in Computational Neuroscience* 2 (2008). Poster presentation.

[49] Z. Zhao and H.-G. Gu. "Transitions between classes of neuronal excitability and bifurcations induced by autapse". *Scientific Reports* 7 (2017), p. 6760.

[50] B. Hutcheon and Y. Yarom. "Resonance, oscillation and the intrinsic frequency preferences of neurons". *Trends in Neurosciences* 23.5 (2000), pp. 216–222.

[51] O. Macherey et al. "A Dual-Process Integrator–Resonator Model of the Electrically Stimulated Human Auditory Nerve". *Journal of the Association for Research in Otolaryngology : JARO* 8 (2007), pp. 84–104.

[52] R. Muresan and C. Savin. "Resonance or Integration? Self-Sustained Dynamics and Excitability of Neural Microcircuits". *Journal of Neurophysiology* 97.3 (2007), pp. 1911–1930.

[53] J. Mitry and M. Wechselberger. "Folded saddles and faux canards". *SIAM Journal on Applied Dynamical Systems* 16.1 (2017), pp. 546–596.

[54] J. Sieber et al. "Experimental Continuation of Periodic Orbits through a Fold". *Physical Review Letters* 100.24 (2008), p. 244101.

[55] G. Marmont. "Studies on the axon membrane. I. A new method". *Journal of Cellular and Comparative Physiology* 34.3 (1949), pp. 351–382.

[56] K. S. Cole. "Ions, potentials and the nerve impulse". *In : Electrochemistry in Biology and Medicine*. Ed. by T. Shedlovsky. 1955, pp. 121–140.

[57] O. P. Hamill et al. "Improved patch-clamp techniques for high-resolution current recording from cells and cell-free membrane patches". *Pflügers Archiv* 391 (1981), pp. 85–100.

[58] E. Neher and B. Sakmann. "The Patch Clamp Technique". *Scientific American* 266.3 (1992), pp. 44–51.

[59] A. A. Sharp et al. "The dynamic clamp: artificial conductances in biological neurons". *Trends in Neurosciences* 16.10 (1993), pp. 389–394.

[60] A. Chizhov et al. "Firing clamp: a novel method for single-trial estimation of excitatory and inhibitory synaptic neuronal conductances". *Frontiers in Cellular Neuroscience* 8 (2014), p. 86.

[61] R. Grashow, T. Brookings, and E. Marder. "Reliable neuromodulation from circuits with variable underlying structure". *Proceedings of the National Academy of Sciences* 106.28 (2009), pp. 11742–11746.

[62] E. Marder et al. "Memory from the dynamics of intrinsic membrane currents". *Proceedings of the National Academy of Sciences* 93.24 (1996), pp. 13481–13486.

[63] J.-M. Goaillard et al. "Slow and persistent postinhibitory rebound acts as an intrinsic short-term memory mechanism". *Journal of Neuroscience* 30.13 (2010), pp. 4687–4692.

[64] H. Ori et al. "Dynamic clamp constructed phase diagram for the Hodgkin and Huxley model of excitability". *Proceedings of the National Academy of Sciences* 117.7 (2020), pp. 3575–3582.

[65] J. B. MacQueen. "Some Methods for Classification and Analysis of MultiVariate Observations". *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. University of California Press, 1967, pp. 281–297.

[66] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". *Applied Statistics* 28.1 (1979), pp. 100–108.

[67] A. K. Jain. "Data clustering: 50 years beyond K-means". *Pattern Recognition Letters* 31.8 (2010), pp. 651–666.

[68] P. beim Graben et al. "Symbolic dynamics of event-related brain potentials". *Physical review E* 62.4 (2000), pp. 5518–41.

[69] P. beim Graben et al. "Optimal estimation of recurrence structures from time series". *Europhysics Letters* 114 (2016), p. 38003.

[70] A. Hutt and P. beim Graben. "Sequences by metastable attractors: interweaving dynamical systems and experimental data". *Frontiers in Applied Mathematics and Statistics* 3 (2017), p. 11.

[71] J. Rinzel and G. B. Ermentrout. "Analysis of neural excitability and oscillations". *In: Methods in neuronal modeling, Second Edition*. Ed. by C. Koch and I. Segev. Vol. 2. 1998, pp. 251–292.

[72] E. Benoît et al. "Chasse au canard". *Collectanea Mathematica* 32.1-2 (1981), pp. 37–119.

[73] M. Desroches, M. Krupa, and S. Rodrigues. "Inflection, canards and excitability threshold in neuronal models". *Journal of Mathematical Biology* 67.4 (2013), pp. 989–1017.

[74] M. Wechselberger, J. Mitry, and J. Rinzel. "Canard theory and excitability". *In: Nonautonomous dynamical systems in the life sciences*. Ed. by P. Kloeden and C. Pötzsche. Vol. **2102**. Lecture Notes in Mathematics. Springer, 2013, pp. 89–132.

[75] V. Krinskiĭ and I. M. Kokoz. "Analysis of the equations of excitable membranes. I. reduction of the hodgkins-huxley equations to a 2d order system". *Biofizika* 18.3 (1973), pp. 506–511.

[76] J. Rinzel. "On repetitive activity in nerve". *Federation proceedings* 37.14 (1978), pp. 2793–2802.

[77] J. Moehlis. "Canards for a reduction of the Hodgkin-Huxley equations". *Journal of Mathematical Biology* 52 (2006), pp. 141–153.

[78] B. Ermentrout. "Type I Membranes, Phase Resetting Curves, and Synchrony". *Neural Computation* 8.5 (1996), pp. 979–1001.

[79] N. Fenichel. "Geometric singular perturbation theory for ordinary differential equations". *Journal of Differential Equations* 31.1 (1979), pp. 53–98.

[80] M. Desroches and V. Kirk. "Spike-Adding in a Canonical Three-Time-Scale Model: Superslow Explosion and Folded-Saddle Canards". *SIAM Journal on Applied Dynamical Systems* 17.3 (2018), pp. 1989–2017.

[81] J. Rinzel. "A Formal Classification of Bursting Mechanisms in Excitable Systems". *International Congress of Mathematicians, Berkeley, California, USA, August 3-11, 1986*. Vol. II. Providence, RI (USA): American Mathematical Society, 1987, pp. 1578–1593.

[82] R. Amir, M. Michaelis, and M. Devor. "Burst Discharge in Primary Sensory Neurons: Triggered by Subthreshold Oscillations, Maintained by Depolarizing Afterpotentials". *The Journal of Neuroscience* 22.3 (2002), pp. 1187–1198.

[83] H. Ori et al. "Dynamic clamp constructed phase diagram for the Hodgkin and Huxley model of excitability". *Proceedings of the National Academy of Sciences* 117.7 (2020), pp. 3575–3582.

[84] A. A. Sharp et al. "Dynamic clamp: computer-generated conductances in real neurons". *Journal of Neurophysiology* 69.3 (1993), pp. 992–995.

[85] M. Wechselberger. "Existence and Bifurcation of Canards in $\mathbb{R}^3$ in the Case of a Folded Node". *SIAM Journal on Applied Dynamical Systems* 4.1 (2005), pp. 101–139.

[86] R. Haiduc. "Horseshoes in the forced van der Pol system". *Nonlinearity* 22.1 (2008), pp. 213–237.

[87] S. J. Schiff et al. "Controlling chaos in the brain". *Nature* 370.6491 (1994), pp. 615–620.

[88] P. Faure and H. Korn. "Is there chaos in the brain? I. Concepts of nonlinear dynamics and methods of investigation". *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie* 324.9 (2001), pp. 773–793.

[89] H. Korn and P. Faure. "Is there chaos in the brain? II. Experimental evidence and related models". *Comptes Rendus Biologies* 326.9 (2003), pp. 787–840.

[90] I. Erchova and D. J. McGonigle. "Rhythms of the brain: An examination of mixed mode oscillation approaches to the analysis of neurophysiological data". *Chaos: An Interdisciplinary Journal of Nonlinear Science* 18.1 (2008), p. 015115.

[91] D. A. McCormick and D. Contreras. "On The Cellular and Network Bases of Epileptic Seizures". *Annual Review of Physiology* 63.1 (2001), pp. 815–846.

[92] M. A. Dichter and G. F. Ayala. "Cellular Mechanisms of Epilepsy: A Status Report". *Science* 237.4811 (1987), pp. 157–164.

[93] A. A. Grace et al. "Dopamine-cell depolarization block as a model for the therapeutic actions of antipsychotic drugs". *Trends in Neurosciences* 20.1 (1997), pp. 31–37.

[94] D. Bianchi et al. "On the mechanisms underlying the depolarization block in the spiking dynamics of CA1 pyramidal neurons". *Journal of Computational Neuroscience* 33 (2012), pp. 207–25.

[95] A. Kuznetsova et al. "Regulation of Firing Frequency in a Computational Model of a Midbrain Dopaminergic Neuron". *Journal of Computational Neuroscience* 28 (2010), pp. 389–403.

[96] K. Tucker et al. "Pacemaker Rate and Depolarization Block in Nigral Dopamine Neurons: A Somatic Sodium Channel Balancing Act". *The Journal of Neuroscience* 32.42 (2012), pp. 14519–31.

[97] K. Qian et al. "Mathematical analysis of depolarization block mediated by slow inactivation of fast sodium channels in midbrain dopamine neurons". *Journal of Neurophysiology* 112.11 (2014), pp. 2779–2790.

[98] M. N. Economo, F. R. Fernandez, and J. A. White. "Dynamic clamp: alteration of response properties and creation of virtual realities in neurophysiology". *Journal of Neuroscience* 30.7 (2010), pp. 2407–2413.

[99] M. S. Espósito et al. "Neuronal differentiation in the adult hippocampus recapitulates embryonic development". *Journal of Neuroscience* 25.44 (2005), pp. 10074–10086.

[100] S. Ge et al. "GABA regulates synaptic integration of newly generated neurons in the adult brain". *Nature* 439.7076 (2006), pp. 589–593.

[101] N. Toni et al. "Neurons born in the adult dentate gyrus form functional synapses with target cells". *Nature Neuroscience* 11.8 (2008), pp. 901–907.

[102] L. O. Wadiche et al. "GABAergic signaling to newborn neurons in dentate gyrus". *Journal of Neurophysiology* 94.6 (2005), pp. 4528–4532.

[103] Y. Tozuka et al. "GABAergic Excitation Promotes Neuronal Differentiation in Adult Hippocampal Progenitor Cells". *Neuron* 47.6 (2005), pp. 803–815.

[104] P. Bielefeld et al. "Insult-induced aberrant hippocampal neurogenesis: Functional consequences and possible therapeutic strategies". *Behavioural Brain Research* 372 (2019), p. 112032.

[105] K. Hüttmann et al. "Seizures preferentially stimulate proliferation of radial glia-like astrocytes in the adult dentate gyrus: Functional and immunocytochemical analysis". *The European Journal of Neuroscience* 18.10 (2003), pp. 2769–2778.

[106] C. L. Indulekha et al. "Seizure induces activation of multiple subtypes of neural progenitors and growth factors in hippocampus with neuronal maturation confined to dentate gyrus". *Biochemical and Biophysical Research Communications* 393.4 (2010), pp. 864–871.

[107] L. A. Mongiat et al. "Reliable activation of immature neurons in the adult hippocampus". *PloS one* 4.4 (2009), e5320.

[108] H. Van Praag et al. "Functional neurogenesis in the adult hippocampus". *Nature* 415.6875 (2002), pp. 1030–1034.

[109] J. M. Parent et al. "Dentate granule cell neurogenesis is increased by seizures and contributes to aberrant network reorganization in the adult rat hippocampus". *Journal of Neuroscience* 17.10 (1997), pp. 3727–3738.

[110] M. Desroches, J. Rinzel, and S. Rodrigues. "Classification of bursting patterns: A tale of two ducks". *PLoS Computational Biology* 18.2 (2022), e1009752.

[111] L. J. Borg-Graham. "Interpretations of Data and Mechanisms for Hippocampal Pyramidal Cell Models". *In: Models of Cortical Circuits*. Ed. by P. S. Ulinski, E. G. Jones, and A. Peters. Boston, MA: Springer US, 1999, pp. 19–138.

[112] N. Jiang et al. "Impaired plasticity of intrinsic excitability in the dentate gyrus alters spike transfer in a mouse model of Alzheimer's disease". *Neurobiology of Disease* 154 (2021), p. 105345.

[113] P. Ambrogini et al. "Morpho-functional characterization of neuronal cells at different stages of maturation in granule cell layer of adult rat dentate gyrus". *Brain Research* 1017.1-2 (2004), pp. 21–31.

[114] M. N. Nenov et al. "Cognitive enhancing treatment with a PPARγ agonist normalizes dentate granule cell presynaptic function in Tg2576 APP mice". *Journal of Neuroscience* 34.3 (2014), pp. 1028–1036.

[115] C. Monier et al. "Orientation and direction selectivity of synaptic inputs in visual cortical neurons: a diversity of combinations produces spike tuning". *Neuron* 37.4 (2003), pp. 663–680.

[116] A. Chizhov and L. Graham. "A strategy for mapping biophysical to abstract neuronal network models applied to primary visual cortex". *PLoS Computational Biology* 17.8 (2021), e1009007.

[117] A. Pokrovskii. "Effect of synapse conductivity on spike development". *Biofizika* 23.4 (1978), 649–653.

[118] O. Shriki, D. Hansel, and H. Sompolinsky. "Rate models for conductance-based cortical neuronal networks". *Neural Computation* 15.8 (2003), pp. 1809–1841.

[119] A. Destexhe and T. Bal, eds. *Dynamic-clamp: From principles to applications*. New York: Springer, 2009.

[120] L. J. Graham and A. Schramm. "In Vivo Dynamic-Clamp Manipulation of Extrinsic and Intrinsic Conductances: Functional Roles of Shunting Inhibition and IBK in Rat and Cat Cortex". *In: Dynamic-clamp: From principles to applications*. Ed. by T. Bal and A. Destexhe. New York: Springer, 2009.

[121] E. Smirnova et al. "The domain of neuronal firing on a plane of input current and conductance". *Journal of Computational Neuroscience* 39.2 (2015), pp. 217–233.

[122] F. R. Fernandez and J. A. White. "Reduction of spike afterdepolarization by increased leak conductance alters interspike interval variability". *The Journal of Neuroscience* 29.4 (2009), pp. 973–986.

[123] F. R. Fernandez and J. A. White. "Gain control in CA1 pyramidal cells using changes in somatic conductance". *The Journal of Neuroscience* 30.1 (2010), pp. 230–241.

[124] P. Mishra and R. Narayanan. "Heterogeneities in intrinsic excitability and frequency-dependent response properties of granule cells across the blades of the rat dentate gyrus". *Journal of Neurophysiology* 123.2 (2020), pp. 755–772.

[125] D. G. Amaral, H. E. Scharfman, and P. Lavenex. "The dentate gyrus: fundamental neuroanatomical organization (dentate gyrus for dummies)". *In: The Dentate Gyrus: A Comprehensive Guide to Structure, Function, and Clinical Implications*. Ed. by H. E. Scharfman. Vol. 163. Progress in Brain Research. Elsevier, 2007, pp. 3–790.

[126] H. Ori, E. Marder, and S. Marom. "Cellular function given parametric variation in the Hodgkin and Huxley model of excitability". *Proceedings of the National Academy of Sciences* 115.35 (2018), E8211–E8218.

[127] D. Levenstein, G. Buzsáki, and J. Rinzel. "NREM sleep in the rodent neocortex and hippocampus reflects excitable dynamics". *Nature Communications* 10 (2019), p. 2478.

[128] J. Hesse et al. "Temperature elevations can induce switches to homoclinic action potentials that alter neural encoding and synchronization". *Nature Communications* 13 (2022), p. 3934.

[129] V. Sip et al. "Characterization of regional differences in resting-state fMRI with a data-driven network model of brain dynamics". *Science Advances* 9 (2023), eabq7547.

[130] J. Ladenbauer et al. "Inferring and validating mechanistic models of neural microcircuits based on spike-train data". *Nature Communications* 10 (2019), p. 4933.

[131] L. Renson et al. "Application of control-based continuation to a nonlinear structure with harmonically coupled modes". *Mechanical Systems and Signal Processing* 120 (2019), pp. 449–464.

[132] R. M. Neville et al. "Shape Control for Experimental Continuation". *Physical Review Letters* 120.25 (2018), p. 254101.

[133] I. Panagiotopoulos, J. Starke, and W. Just. "Control of collective human behavior: Social dynamics beyond modeling". *Physical Review Research* 4.4 (2022), p. 043190.

[134] A. P. Willis et al. "Surfing the edge: using feedback control to find nonlinear solutions". *Journal of Fluid Mechanics* 831 (2017), 579–591.

[135] I. de Cesare et al. "Control-Based Continuation: A New Approach to Prototype Synthetic Gene Networks". *ACS Synthetic Biology* 11.7 (2021), pp. 2300–2313.

[136] M. Blyth et al. "Numerical methods for control-based continuation of relaxation oscillations". *Nonlinear Dynamics* 111 (2023), pp. 1–18.

[137] P. C. Schwindt and W. E. Crill. "Properties of a persistent inward current in normal and TEA-injected motoneurons." *Journal of Neurophysiology* 43.6 (1980), pp. 1700–24.

[138] H. M. Fishman and R. I. Macey. "The N-Shaped Current-Potential Characteristic in Frog Skin: I. Time Development during Step Voltage Clamp". *Biophysical Journal* 9.2 (1969), pp. 127–139.

[139] D. Johnston, J. J. Hablitz, and W. A. Wilson. "Voltage clamp discloses slow inward current in hippocampal burst-firing neurones". *Nature* 286.5771 (1980), pp. 391–393.

[140] M. R. Blatt. "Potassium-dependent, bipolar gating of K^+ channels in guard cells". *Journal of Membrane Biology* 102 (1988), pp. 235–246.

[141] K. Vervaeke et al. "Contrasting Effects of the Persistent Na+ Current on Neuronal Excitability and Spike Timing". *Neuron* 49.2 (2006), pp. 257–70.

[142] J. Rinzel. "A Formal Classification of Bursting Mechanisms in Excitable Systems". *Mathematical Topics in Population Biology, Morphogenesis and Neurosciences: Proceedings of an International Symposium held in Kyoto, November 10-15, 1985*. Ed. by E. Teramoto and M. Yamaguti. Vol. 71. Lecture Notes in Biomathematics. Springer Berlin Heidelberg, 1987, pp. 267–281.

[143] D. Amakhin et al. "Insertion of Calcium-Permeable AMPA Receptors during Epileptiform Activity In Vitro Modulates Excitability of Principal Neurons in the Rat Entorhinal Cortex". *International Journal of Molecular Sciences* 22.22 (2021), p. 12174.

[144] I. Magrans de Abril, J. Yoshimoto, and K. Doya. "Connectivity inference from neural recording data: Challenges, mathematical bases and research directions". *Neural Networks* 102 (2018), pp. 120–137.

[145] J. Sieber and B. Krauskopf. "Control based bifurcation analysis for experiments". *Nonlinear Dynamics* 51 (2008), pp. 365–377.

[146] K. Pyragas. "Continuous control of chaos by self-controlling feedback". *Physics Letters A* 170.6 (1992), pp. 421–428.

[147] K. Pyragas. "Control of Chaos via an Unstable Delayed Feedback Controller". *Physical Review Letters* 86.11 (2001), pp. 2265–2268.

[148] C. Marschler et al. "Implicit Methods for Equation-Free Analysis: Convergence Results and Analysis of Emergent Waves in Microscopic Traffic Models". *SIAM Journal on Applied Dynamical Systems* 13.3 (2014), pp. 1202–1238.

[149] I. Panagiotopoulos et al. "Continuation with Non-invasive Control Schemes: Revealing Unstable States in a Pedestrian Evacuation Scenario". *SIAM Journal on Applied Dynamical Systems* 22.1 (2023), pp. 1–36.

[150] C. G. Broyden. "A Class of Methods for Solving Nonlinear Simultaneous Equations". *Mathematics of Computation* 19.92 (1965), pp. 577–593.

[151] E. M. Izhikevich and R. FitzHugh. "FitzHugh-Nagumo model". *Scholarpedia* 1.9 (2006), p. 1349.

[152] S. Morfu et al. "A nonlinear electronic circuit mimicking the neuronal activity in presence of noise". *2013 22nd International Conference on Noise and Fluctuations (ICNF)*. 2013, pp. 1–4.

[153] D. Gottlieb and S. A. Orszag. *Numerical Analysis of Spectral Methods*. Society for Industrial and Applied Mathematics, 1977.

[154] Z. Peng et al. "Comparisons between harmonic balance and nonlinear output frequency response function in nonlinear system analysis". *Journal of Sound and Vibration* 311.1 (2008), pp. 56–73.

[155] G. M. Shepherd. "The Synaptic Organization of the Brain". Oxford University Press, 2004.

[156] N. Miyasaka et al. "From the Olfactory Bulb to Higher Brain Centers: Genetic Visualization of Secondary Olfactory Pathways in Zebrafish". *Journal of Neuroscience* 29.15 (2009), pp. 4756–4767.

[157] G. Lepousez, A. Nissant, and P.-M. Lledo. "Adult neurogenesis and the future of the rejuvenating brain circuits". *Neuron* 86.2 (2014), pp. 387–401.

[158] O. Gschwend et al. "Neuronal pattern separation in the olfactory bulb improves odor discrimination learning". *Nature Neuroscience* 18.10 (2015), pp. 1474–1482.

[159] H. K. Kato et al. "Parvalbumin-expressing interneurons linearly transform cortical responses to visual stimuli". *Neuron* 73.1 (2013), pp. 159–170.

[160] S. D. Burton and N. N. Urban. "Greater excitability and firing irregularity of tufted cells underlies distinct activation of the olfactory bulb output". *PLoS Biology* 13.8 (2015), e1002318.

[161] A. Li, D. H. Gire, and T. Bozza. "The olfactory bulb encodes odor valence and type in parallel pathways". *Cell Reports* 22.9 (2018), pp. 2634–2645.

[162] A. M. Boyd et al. "Cortical feedback control of olfactory bulb circuits". *Neuron* 76.6 (2012), pp. 1161–1174.

[163] K. Mori and H. Sakano. "How Is the Olfactory Map Formed and Interpreted in the Mammalian Brain?" *Annual Review of Neuroscience* 34.1 (2011), pp. 467–499.

[164] D. D. Stettler and R. Axel. "Representations of odor in the piriform cortex". *Neuron* 63.6 (2009), pp. 854–861.

[165] M. Wachowiak. "All in a sniff: olfaction as a model for active sensing". *Neuron* 71.6 (2011), pp. 962–973.

[166] T. Cover and P. Hart. "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27.

[167] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

[168] D. W. Aha, D. Kibler, and M. K. Albert. "Instance-based learning algorithms". *Machine Learning* 6.1 (1991), pp. 37–66.

[169] C. Cortes and V. Vapnik. "Support-vector networks". *Machine learning* 20.3 (1995), pp. 273–297.

[170] C. M. Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. Information Science and Statistics. Springer, 2006.

[171] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[172] S. Ruder. "An overview of gradient descent optimization algorithms". arXiv e-print 1609.04747. 2016.

[173] R. Kohavi. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, 1137–1143.

[174] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.

[175] B. Efron. "Bootstrap Methods: Another Look at the Jackknife". *The Annals of Statistics* 7.1 (1979), pp. 1–26.

[176] T. J. DiCiccio and B. Efron. "Bootstrap confidence intervals". *Statistical Science* 11.3 (1996), pp. 189–228.

[177] M. Chernick and R. LaBudde. *An Introduction to Bootstrap Methods with Applications to R*. Wiley, 2014.

[178] H. Han, W. Wang, and B. Mao. "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning". *In: Advances in Intelligent Computing, International Conference on Intelligent Computing, ICIC 2005, Hefei, China, August 23-26, 2005, Proceedings, Part I*. Ed. by D.-S. Huang, X.-P. Zhang, and G.-B. Huang. Vol. 3644. Lecture Notes in Computer Science. 2005, pp. 878–887.

[179] T. Maciejewski and J. Stefanowski. "Local neighbourhood extension of SMOTE for mining imbalanced data". *In: Proceeding of the 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*. Ed. by N. Chawla, I. King, and A. Sperduti. 2011, pp. 104–111.

[180] G. Batista, R. Prati, and M.-C. Monard. "A Study of the Behavior of Several Methods for Balancing machine Learning Training Data". *ACM SIGKDD Explorations Newsletter* 6.1 (2004), pp. 20–29.

[181] N. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[182] R. S. Olson et al. "Automating Biomedical Data Science Through Tree-Based Pipeline Optimization". *In: Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part I*. Ed. by G. Squillero and P. Burelli. Springer International Publishing, 2016, pp. 123–137.

[183] R. S. Olson et al. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science". *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. Ed. by T. Friedrich. GECCO '16. Denver, Colorado, USA: ACM, 2016, pp. 485–492.

[184] T. T. Le, W. Fu, and J. H. Moore. "Scaling tree-based automated machine learning to biomedical big data with a feature set selector". *Bioinformatics* 36.1 (2019), pp. 250–256.

[185] A. Wagemakers et al. "Building electronic bursters with the Morris–Lecar neuron model". *International Journal of Bifurcation and Chaos* 16.12 (2006), pp. 3617–3630.