UNIVERSITÀ
DI PAVIA

Bachelor Thesis

A computational perspective on Purkinje cell representation: multi-compartmental to reduced models in point-neuron cerebellar circuits

Arrigoni Anna Clara
Artificial Intelligence

Academic Year 2024/2025

Bachelor in Artificial Intelligence
Pavia 2025

Bachelor in Artificial Intelligence

Arrigoni Anna Clara

A computational perspective on Purkinje cell representation: multi-compartmental to reduced models in point-neuron cerebellar circuits

Supervisors: **Casellato Claudia (Supervisor)**
Benozzo Danilo (Co-supervisor)
Rizza Martina (Co-supervisor)

© Arrigoni Anna Clara, 2025

Abstract

The Neuro-AI pathway explores the mutual reinforcement between neuroscience and Artificial Intelligence (AI). In this framework, the internal complexity of individual neurons plays a key role in informing the design of AI models: the computational richness of single neurons can inspire the development of more efficient and powerful systems, drawing inspiration from biological neural mechanisms.
This thesis focuses on neurons and microcircuits of the cerebellar regions, as the cerebellum is involved in a wide range of functions, encompassing sensory-motor tasks, high-level cognitive processes, and emotional regulation. The main objective of this work is to develop and test an automatic simplification pipeline that reduces detailed multi-compartmental neuron models into more compact few-compartmental models. This approach is mainly applied to the cerebellar Purkinje cell, which acts as a complex perceptron and represents the main output of the cerebellar cortex. To this end, the NEAT Python library, which is part of the NEST simulator ecosystem and tailored for the study, simulation and simplification of morphological neuron models, is employed. Multiscale tools are also integrated, from the Brain Scaffold Builder (BSB) framework to the NEST and NEURON simulators. The next phase will involve scaling up to circuit-level analyses to assess the impact on network dynamics. Finally, the thesis outlines potential future directions, including applications in the study of brain diseases, for example in autistic cerebellar models, to gain deeper insights into the underlying pathological mechanisms.

Keywords

cerebellum, brain modeling, computational neuroscience, purkinje cells, multi-compartmental models, brain scaffold builder, neuron model simplification

Contents

Chapter 1
Introduction

The cerebellum, long regarded as primarily responsible for motor coordination, has emerged as a crucial player in a much broader range of brain functions, including cognition, emotion, and perception. This growing body of evidence has drawn increasing attention from the field of computational neuroscience, where modeling brain structures can offer deep insights into both normal and pathological conditions.
In this context, the present work seeks to bridge anatomical and functional understanding of the cerebellum with computational approaches, both considering point-neuron network models of the cerebellar cortex and emphasizing the role of the Purkinje cell by adopting both multi-compartmental and few-compartmental approaches, and highlighting possible future research perspectives in the specific pathological case of the autism spectrum disorder.
By focusing on biological relevance and simulation frameworks, the thesis aims to clarify the importance and potential of the cerebellum in modern neuroscience.
The following sections outlines the specific goals, framework and motivations of this work, followed by a brief outline of the content of this document.

1.1 Aim of the thesis

The present document aims to provide a collection of some solid evidence on the importance of the cerebellum, given its contributions to numerous brain functions, in the field of computational neuroscience, guiding the work from the standpoint of the Neuro-AI approach. This thesis also aims to report the results of a personal work on reconstructing and simulating the cerebellar network with a point-neuron approach and on few-compartmental models, obtained through a reduction of multi-compartmental models, using the NEAT Python library, of the Purkinje cell, to ultimately offer some future research perspectives, specifically in the IB2 KO autistic case in mice models.

1.2 The current work within the AI landscape

Given the very nature of my Bachelor's degree, before presenting an overview of the content of the current document, it is indeed adequate to understand the relation it has with the field of Artificial Intelligence (AI).

Historically, Artificial Intelligence has evolved along two major paradigms. On one side the data-driven, computational approach, primarily embodied by machine learning, aims to produce intelligent behavior by extracting patterns from large datasets through complex statistical computations, while on the other hand, a biologically inspired line of re-

search seeks to understand and replicate the actual structures and mechanisms of the brain, under the hypothesis that intelligence arises from the physical architecture and dynamics of the nervous system. The internal complexity within individual neurons can in fact play a crucial role in designing AI models, as the computational richness of single neurons can lead to more efficient and powerful AI systems, drawing inspiration from biological neural mechanisms [24]. This latter perspective includes the research branch of brain modeling, and more specifically, the study of neuron models such as the multi-compartmental, few-compartmental, and point-neuron ones, like those of Purkinje cells featured in the current work. While there is no agreed upon definition of AI, these two main conceptions remain central in AI discourse since the rise of the field, as scholars from all times showed perspectives aligned either with one or the other line of thought. For example, Marvin Minsky quoted that AI is "the science of making machines do things that would require intelligence if done by men" [44], siding more with the "reproducing intelligence" paradigm, while Nils John Nilsson, stated that "Artificial intelligence is that activity devoted to making machines intelligent" [52], aligning more with the "producing intelligence" approach. The latter approach leans more towards emulation rather than mere simulation of the brain: it strives not just to mimic external behavior, but to reproduce the internal functional structure that gives rise to intelligence, and the ambit of brain modeling aligns more with this approach.

As a final remark, we can underline the distinction between general and narrow AI and link it to the current discourse: while the expression "general AI" refers to systems able to carry out any task as well as (or even better than) a human would, "narrow AI" refers to systems that can carry out only one or a limited number of tasks, even if they achieve their goals perfectly [58]. At the moment there are no examples of general AI systems, but many of narrow AI, for example robots that play chess, large language models, recommendation algorithms and so on. Current neural models pertain to this definition, as they are limited in their nature and programmed to simulate neural electrical activity. Nonetheless it is possible that they will provide valuable insights and contribute to a broader, more general understanding of intelligence in the future.

1.3 Thesis outline

After the current introductory section, the rest of the thesis is organized as follows:

Chapter 2 is entirely dedicated to the cerebellum: in this chapter the structure and functions of the aforementioned organ are analyzed, including the anatomical, biophysiological and electrical properties that are most relevant for the matter of this thesis. A review of the current state of modeling this brain region is also offered.

Chapter 3 is concerned with presenting the Brain Scaffold Builder (BSB) framework as a state-of-the-art tool in terms of modeling this brain region, along with the other computational tools adopted in my project work: NEURON, NEST and NEAT. Their main features and characteristics are pinpointed.

Chapter 4 ties the strings together by presenting a practical modeling work carried out during this academic year, focusing on the mouse cerebellar cortex reconstruction and simulation within the BSB framework with a point-neuron approach, alongside an in-depth focus on both regular multi-compartmental and reduced multi-compartmental or few-compartmental models of the Purkinje cell.

Chapter 5 presents some insights on the social and medical implications of research on the matter of cerebellar modeling, presenting some ongoing large-scale projects and with a specific focus on the autistic pathological case: a general overview of the condition is provided, with special attention to the case of the cerebellar network

of IB2 KO autistic mice and its properties. Additionally, a brief section on animal testing and experiments is included.

Chapter 6 contains conclusions, addresses limitations of my work and possible future research on the matter of my thesis.

Chapter 2
The cerebellum

The cerebellum is named after the Latin word *cerebellum*, which means "little brain". Despite its reduced measures compared to the rest of the brain, this organ contains approximately 4.3 times the number of neurons contained in the cerebral cortex and more than half of all cerebral neurons [35]. This datum alone suggests the relevance of this organ in defining the neural framework of an individual.
This chapter addresses the centrality and importance of the cerebellum in modeling, first by introducing some anatomical and biophysiological evidence that will come in handy later, and in a second moment by presenting its most basic functions, namely muscular tone maintenance, posture and balance maintenance and the coordination of movements, and other relevant mechanisms it is involved in, such as attention, prediction, and emotional regulation.
Finally, an overview of its importance in modeling, through the literature in computational neuroscience dedicated to this brain region, is provided.

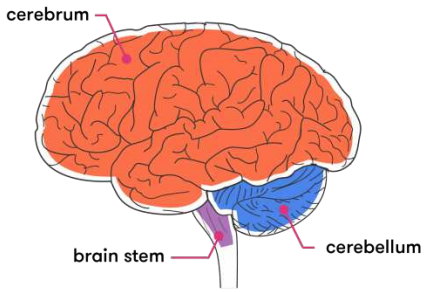


Figure 2.1: Representation of the size and location of the cerebellum

2.1 Position and developement of the cerebellum

As showed in **Figure 2.1**, the cerebellum is located at the back of the brain stem, making it part of the hindbrain. It develops from the rhombencephalon, one of the three components that form during embryonic brain development, along with the mesencephalon (which corresponds to the midbrain) and the prosencephalon (which corresponds to the forebrain) .
More specifically, the cerebellum develops from one of the two secondary vesicles of the rhombencephalon: the metencephalon, which also gives rise to the pons. Spatially, the

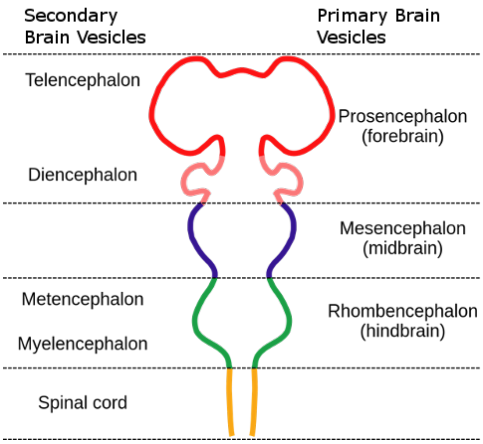


Figure 2.2: A schematic representation of the vesicles that form during brain development

cerebellum develops posteriorly from the pons. For the sake of completeness, the other vescicle is called the myelencephalon and it gives rise to the medulla oblungata [60, 28]. **Figure 2.2** represents these vesicles in a schematic way.

2.2 Components of the cerebellum

Let us now investigate the components that constitute the cerebellum. There are three main areas: the anterior lobe, located at the top of the organ, the posterior lobe, which is the largest component and extends from the top to the bottom of the cerebellum, and the floccular area (also called the flocculonodular lobe), the smallest component, "hidden" behind the posterior lobe. Each of these parts has distinct functions, developmental processes, and connections, which we will investigate.
Primitive in its development is the floccular area, which is also called archicerebellum or primitive cerebellum because of this fact. It is mainly responsible for balance, especially concerning the head and the eyes. The anterior lobe, also known as paleocerebellum, is primarly involved in maintaining muscular tone. Finally, the posterior lobe, or neocerebellum, plays a dominant role in the coordination of movements.
There are two major fissures separating these areas, namely the primary fissure between the anterior and posterior lobes and the posterolateral (or dorsolateral) fissure between the posterior and flocculonodular lobes [63]. When looking at the cerebellum from the back instead of from the side, we are able to divide functional properties from left to right instead of from top to bottom. In this view we can notice the presence of a longitudinal depression in the center of the organ, surrounded by more prominent lateral regions. The cerebellar cortex present in the depression is called the vermis, and on either side lies a functionally distinct area, called the paravermal area (or intermediate zone) which is part of the larger cerebellar hemispheres (left and right). A lesion in the cerebellum typically affects the same side of the body (ipsilateral effect), unlike lesions in the cerebrum, which usually have contralateral effects. Both the vermis and paravermal areas contain a topographical representation of the body, used to collect and constantly update sensory information. Specifically, sensory inputs from the limbs are processed in the paravermal area, while those from the trunk (neck, shoulders, trunk, hips) are processed in the vermis. As a result, the vermis is mainly involved in motor control related to axial musculature (neck-shoulder-trunk-hips). Lesions in this area typically cause issues with these movements. Meanwhile, the paravermal region is involved in the control of finer movements of the limbs, hands, and feet, which are more distal and require more precision. Lesions

here affect those specific motor functions. This homunculus-like model holds for both the paleo- and neocerebellum, but it is not present in the hemispheres [34].
Figure 2.3 offers a simple representation of these areas.

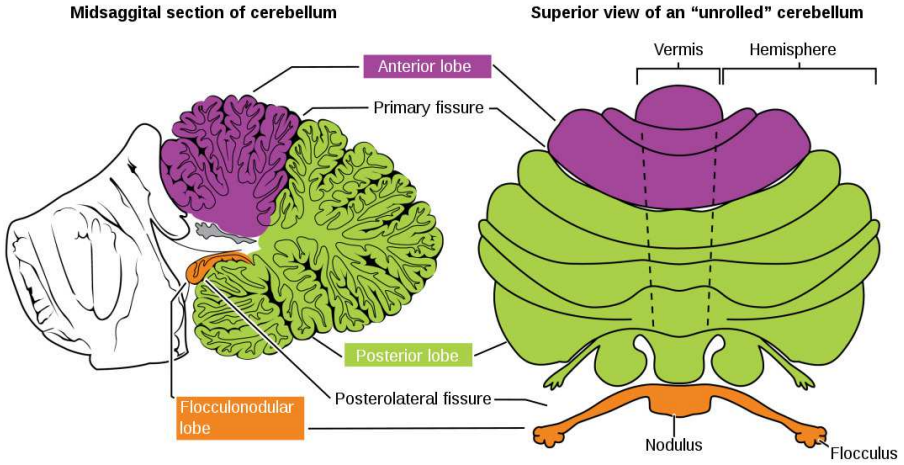


Figure 2.3: The components of the cerebellum

2.3 Connections of the cerebellum

The cerebellar cortex is the region where the neuronal cell bodies lie. It is also referred to as the cerebellar gray matter, or surface gray matter. In contrast, the deep cerebellar nuclei are located in the inner part of this organ. These nuclei are clusters of gray matter (deep gray matter) embedded within the white matter, which is composed of axons and fibers, mainly. Numerous fibers bring information into the cerebellum. These fibers originate from various sources such as the brain stem, higher brain areas, and the spinal cord. Each group of fibers has a specific name, but in this document we will refer to mainly mossy and climbing fibers. As a matter of fact, the majority of afferent fibers terminate as mossy fibers, synapsing onto granule cells in the cerebellar cortex, whereas a smaller population of climbing fibers, arising exclusively from the inferior olivary nucleus, form powerful synaptic connections directly with Purkinje cells. These fibers are excitatory and their primary neurotransmitter is glutamate [54].

2.3.1 Internal cerebellar structure

From the previous paragraph it should now be clear that input to the cerebellum comes from the inferior olivary nucleus (via climbing fibers) and from all the mossy fibers. Hence, in this section, we will focus on internal processing and output of the cerebellum. Both mossy and climbing fibers need to pass through the deep cerebellar nuclei before reaching the cerebellar cortex. These fibers release only excitatory neurotransmitters, among which mainly glutamate, stimulating the deep nuclei.
There are three layers in the cerebellar cortex. Let us explore what happens in each of them. The outermost layer is called the molecular layer. Here, climbing fibers reach the dendrites of Purkinje cells, forming strong excitatory one-to-one connections. Also, in this layer, the axons of the granule cells bifurcate in parallel structures (forming the so called parallel fibers) with respect to the cortex orientation, connecting via excitatory synapses to millions of Purkinjie cells dendrites and contributing to creating some white matter. Ultimately, in this layer one could find basket and stellate cells, which are stimulated by parallel fibers, and in turn inhibit Purkinje cells through the GABA neurotransmitter. The middle layer is called the Purkinje layer, consisting of the large Purkinje cell bodies. Their

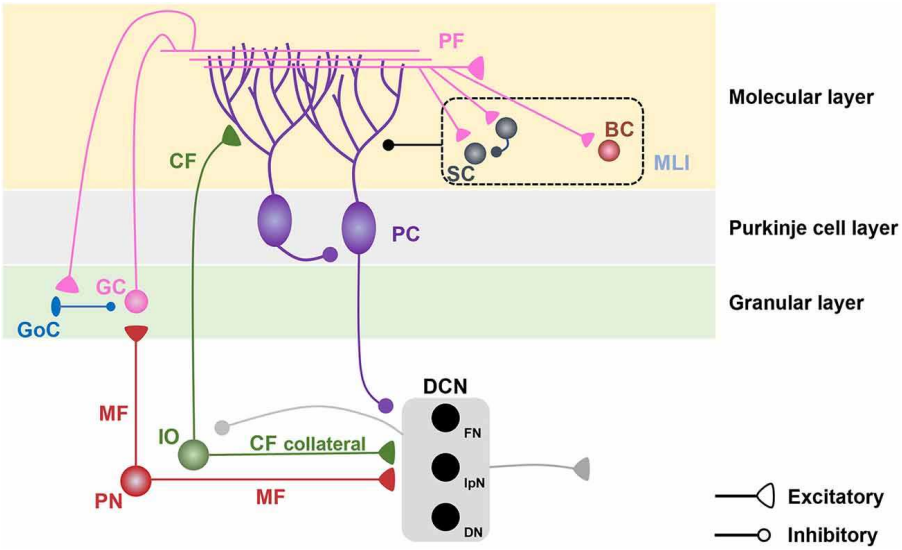


Figure 2.4: A schematic representation of the internal structure of the cerebellum

axons extend downward to the deep cerebellar nuclei forming an inhibitory connection by producing GABA, while their dendrites go up to the molecular layer, where central Purkinje cells will be directly excited by parallel fibers while neighboring Purkinje cells will be indirectly inhibited by stellate and basket cells, so that the signal is sharpened by enhancing contrast, making the activity more focused and less diffuse. The innermost layer is called the granular layer and it contains granule cells that receive the input of mossy fibers. Each granule cell has multiple dendrites (typically four), and each dendrite forms three synapses, two excitatory and one inhibitory, allowing it to integrate multiple signals. The axons of granule cells ascend to the molecular layer to become parallel fibers. This layer also includes Golgi cells, which are excited by the parallel fibers and inhibit granule cells via GABA, creating an autoinhibitory loop that prevents excessive firing. Mossy fibers can also directly activate Golgi cells, further contributing to the inhibition of granule cells. Finally, the output of the system is provided by the deep cerebellar nuclei, excited by mossy and climbing fibers and inhibited by Purkinje cells [39, 72].
The **Figure 2.4** represents the layers and the cells they contain in a schematic way.

2.3.2 Cerebellar outer connections and dynamics

Recall that the cerebellum is composed of three main areas: the archicerebellum or vestibular cerebellum (connected with the vestibular nuclei), the paleocerebellum or spinal cerebellum (connected with the spinal cord) and the neocerebellum or cerebro-cerebellum (connected with the cerebral cortex) [34].
Starting with the vestibular cerebellum, its connections originate from the vestibular nuclei, located in the inner ear, and part of a wider vestibular apparatus. These nuclei are responsible for balance, particularly in relation to head positioning. Sensory input comes from two key organs, the utricle and the saccule, which detect horizontal and vertical movements, respectively. Both organs contain sensory epithelium and by moving our head we displace some fluid into the semi circular canals attached to them, which will then stimulate the epithelium, sending action potentials along fibers that either enter the cerebellum directly or pass through the vestibular nuclei first, ultimately reaching the flocculonodular lobe. All of these fibers are mossy fibers. From the deep cerebellar nuclei the output goes back to the vestibular nuclei through cerebello-vestibular fibers. From there, signals may descend via the vestibulospinal tract to regulate the muscle tone and maintain balance for extensor muscles, or through the reticulospinal tract to influence flexor

muscles, or ascend via the medial longitudinal fasciculus to coordinate eye movements during head motion [33, 39].
Regarding the spinal cerebellum, the vermal and paravermal areas receive sensory information from the spinal chord. For lower limbs and the lower trunk, information comes from the muscle spindle, the Golgi tendon organ and other receptors in ligaments and joints communicating with the spinal chord, which travel to the cerebellum through the dorsal spinocerebellar tract via the inferior cerebellar peduncle. In the case of upper limb proprioception, a similar process occurs but involves a relay in the cuneate nucleus before entering the cerebellum through the cuneocerebellar pathway. Additionally, some fibers arise more ventrally in the spinal cord and cross over contralaterally before reaching the cerebellum through the superior peduncle as part of the ventral spinocerebellar tract. These fibers then cross again upon entry, ultimately projecting to the correct side. In summary , information about current body movements enters the cerebellum and information about the final move goes returns to the spinal chord, forming the basis of the spinal cerebellar system.
It is now relevant to distinguish between the types of deep cerebellar nuclei involved in output. From the most lateral to the most medial, we find the dentate nucleus, which is associated with the neocerebellum; the emboliform and globose nuclei, collectively known as the interposed nuclei, connected to the spinal cerebellum; and finally the fastigial nucleus, which links to the vestibular cerebellum. The output of the spinal cerebellum comes from the interposed nuclei and reaches the red nucleus via cerebello-rubral fibers, then proceeds to the thalamus. From there, thalamo-cortical projections influence the motor, premotor, and somatosensory cortices. Corrections to movement are implemented either through the corticospinal tract or directly from the red nucleus via rubrospinal fibers [38, 39].

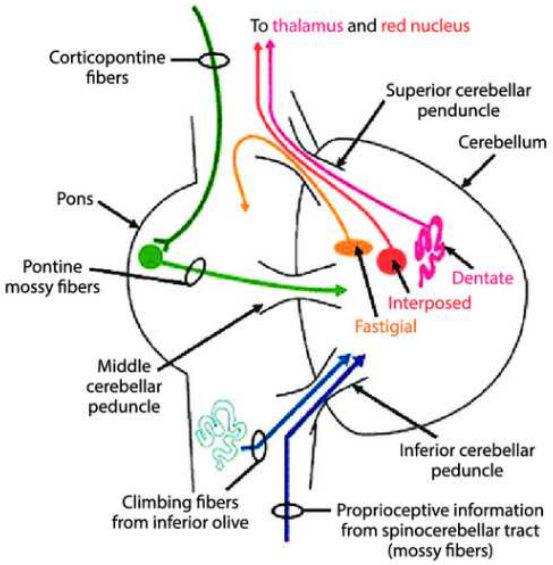


Figure 2.5: A schematic representation of the outer connections of the cerebellum

Finally, as for the cerebro-cerebellar connections, we focus on the hemispheres of the cerebellum. The connections of our interest start from the pontine nuclei, located in the pons, which receives projections from various cortical areas including the premotor, motor, and somatosensory cortices. Pontine fibers cross to the opposite side and enter the middle cerebellar prudencle (following the cortico-ponto-cerebellar pathway) and go into the cerebellum, where, after some processing, the resulting output from the dentate nucleus travels through dento-rubral-thalamic or dento-thalamic fibers and returns to the motor regions

of the cerebral cortex. This establishes a closed-loop system responsible for movement planning and intended motor actions [54, 11].
The **Figure 2.5** represents the outer connections of the cerebellum in a schematic way.
In conclusion, balance is primarily managed by the vestibular cerebellum, actual movement adjustment by the spinal cerebellum, and the planning and intention of movement by the cerebro-cerebellar connections. As a result, the cerebellum is a master controller of movement, posture mantainance and muscolar tone maintainance. However, as stated previously, movement is not the only function handled by the cerebellum. In the following section we are also going to explore some of the other functions it deals with.

2.4 The many functions of the cerebellum

This section investigates the main and many functions of the cerebellum, from the sensory-motor related ones to those of more novel discovery, related to high level cognitive functions, emotions, and social attitudes.

2.4.1 The role of the cerebellum in the motor system

In this paragraph we analyze how the cerebellum interacts with the premotor and motor cortexes and the supplementary motor cortex in the pipeline of movement making. Movement planning originates in the prefrontal cortex and is then relayed to the premotor and supplementary motor areas. These areas, in turn, interact with the basal ganglia, which refine the motor program. The finalized plan is then returned via the thalamus to the premotor area, and subsequently transmitted to the primary motor and somatosensory cortices. From here, corticospinal fibers carry the signal downward to initiate muscle contractions. The cerebellum, however, plays a crucial, complementary role. To execute a movement accurately, the body's initial position must be known. The central nervous system knows it, because this information is present in the cerebellum, which (unconciously) collects information from muscles, joints, tendons, ligaments, etc. so that it always has a sort of motor picture of the body. The cerebellum knows the initial position because it is updated all the time by the local motor system and, as a consequence, it can guide us in our movements. This is due to the fact that intendend movements (plans) are also fed to the cerebellum and not basal ganglia exclusively. By receiving a copy of the intended plan and the current state of the body, the organ makes a comparison the two and detects the differences between them, providing information on how to do a movement back to the central nervous system. Then, the motor program is passed down the cortical spinal fibers that fire to make the muscles contract, but before that, after all central processing has been done, the cerebellum does a double check through a copy of the finalized plan of the initiated movement to see if the program is adequate or not, and only then the movement starts. All the information generated by movements, like the firing obtained through the contraction of muscles and fibers, is again given to the cerebellum to see how the progression of the movement is going. This way a continuous update is possible. This mechanism also makes the cerebellum a predictive system as, from the information received (for example, the force applied to complete a movement or the angulation of an articulation), it can guess what will happen in the future, as in guessing the final position of your body or by iteratively fixing and correcting a movement [54, 31, 38].

2.4.2 Other functions of the cerebellum

As previously mentioned, the cerebellum has recently emerged as a key structure involved in a wide range of non-motor functions, including cognitive, emotional, and even social processes. This paragraph offers a brief ovreview of some literature that supports this claim, underlining the centrality in modeling this organ in the neural framework of an

individual.

It has been demnonstrated that patients with cerebellar damage often exhibit impairments in working memory, abstract reasoning, language fluency, and social cognition [2, 3, 67].
One of the most compelling models describing cerebellar involvement in cognition and emotion is the concept of the Cerebellar Cognitive Affective Syndrome (CCAS). This syndrome is characterized by deficits in executive functions (such as planning, set-shifting, abstract reasoning), language (agrammatism, anomia), spatial cognition, and affect regulation (ranging from blunted affect to disinhibition), and occurs following damage to the posterior lobe of the cerebellum, particularly in the lateral hemispheres [59].
In particular, sensitive periods in cerebellar development, an early stage in life when the brain is uniquely plastic to intrinsic and extrinsic stimuli and has the ability to create new connections, are crucial to understand cerebellar impairment and how it affects high cognitive level functions and the affective and social spheres [57].
Futhermore, research shows that the cerebellum is involved in social recognition memory and mentalizing, contributing to the understanding of others' intentions. Experimental manipulations in animal models confirm that cerebellar circuits play a role in retrieving socially relevant emotional information, with implications for conditions like autism spectrum disorder. Moreover, the cerebellum contributes to reward processing and expectation and to arousal and autonomic regulation [57]. As a result, some studies argue that the cerebellum has a critical role in human social and emotional learning and that it even covers a prominent role in the learning, origin and advancement of culture [66, 68].
These findings collectively indicate that the cerebellum is deeply embedded in the neural architecture of complex behavior, far beyond its traditional motor domain.

2.5 Modeling the cerebellum

Having traced the cerebellar developmental trajectory, examined its structural organization, extensive outer connectivity and wide-ranging functions, it becomes evident that any attempt to model the brain must give due consideration to this region. Its complexity and integration within broader neural systems make it a critical target for computational approaches, therefore, in the current section, a small review on the attempts of modeling this brain region is given.

Let us start by stressing how the relationship between structure, function and dynamics in brain circuits, the cerebellar cortex in our case, is still poorly understood. A way to deal with this challenge is to integrate experimental procedures with modeling techniques. Computational models can reproduce morphological and structural charactheristics of the networks, alongside functional, physiological and electrical ones, and once properly configured and validated, they can generate some piece of new evidence and test new hypotheses while simulating real networks.
In the existing literature, two main approaches to model neural circuits, more specifically to model each neuron that composes these circuits, arise: on one hand we have multi-compartmental or conductance-based models of neurons, while on the other we have point-neuron models. Multi-compartmental models are detailed copies of real neurons, that aim to reproduce their morphology and activity faithfully by modeling the ionic activity present on the membrane of neural cells (from this standpoint, the name "conductance-based"). Their biological realism does not come without a cost, though. In fact, them being so complex and detailed also makes computational processes involving them long and expensive. Their counterpart, point-neuron models, are on the other hand way more computationally efficient, as they model neurons as if they were single points (ideally representing their soma), but it is now intuitive to notice how their nature makes them less biologically accurate and realistic [30]. The **Figure 2.6** shows a schematic representation of multi-compartmental and point-neuron models of a Purkinje cell.

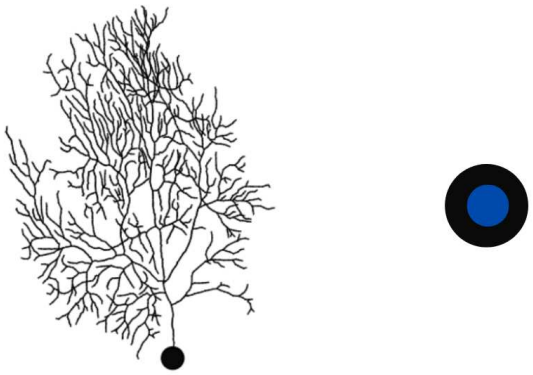


Figure 2.6: A schematic representation of a multi-compartmental model of the Purkinje cell compared to a point-neuron one

Among these models, we are to investigate the ones reproducing the cerebellar cortical microcircuit, the novelties they introduce, and their limitations.
Some network models using ionic conductance-based neurons have been developed (more information in section 3.2.2), but they were limited to the innermost layer of the cortex: the granular layer [62].
Some models that included both the granular and molecular layer altogether were developed in a second moment, but they lost some degree of biological realism as they made use of point-neurons (more information in section 3.3), with a simplified representation of membrane excitability [12]. These models displayed a nice predictive power if validated against some target parameters, however their connectivity was set independently from their morphology, meaning that connections were set randomly without considering the structure of the neurons themselves, therefore preventing a direct link between microcircuit structure, function and dynamics [15]. Furthermore, detailed and multi-compartmental computational models (more information in section 3.2) of the main cerebellar cortical neurons, embedding multiple membrane ionic channels and synaptic receptors, have been developed, tested and validated [15, 40, 41, 55].
Finally a framework for modeling the cortex in its entirety and making use of multi-compartmental model was presented: the Brain Scaffold Builder (BSB) [15], which will be explored more deeply in **Chapter 3**.
Working within this framework, it was possible to generate a realistic connectome of all the layers, following precise connection rules. Furthermore, simulation in different conditions and validation against recordings in vivo was possible, also providing a benchmark for investigation of unknown properties and future research.
In the process, the volume of the network was defined first, along with all the cell types. The BSB then proceeded with cell placement and connectivity rules, reconstructing the microcircuit network. Finally, the BSB was interfaced with the NEURON simulator (see section 3.2) and network activity was simulated in different conditions and the results visualized.
This work lead to the creation of the first detailed model of the cerebellar cortical network for both simulation and prediction of neural activity generated by the input provided by mossy fibers. It was optimized and validated showing compatibility with experimental data, and it is suitable for parameter alteration in pathological cases in terms of neural correlates of behavior.
Comparing past cerebellar models, earlier versions captured specific network properties but lacked realistic multi-compartment neuronal structures, neurotransmitter dynamics, or detailed connectivity rules, while the BSB model integrated the missing elements, although some limitations cannot be negleted. For example, some unexplored parameter

combinations could be effective, some structural data were missing and the model is much smaller (1000 times) than the actual rodent cortex, impacting on signal propagation and microzones properties. Still, the model overall underlines the importance of geometrical organization of the cells for realistic copies, offering a benchmark for future research even for cerebellar alterations and simulations.

Having established a comprehensive understanding of the cerebellum, from its developmental origins, to its structural and functional complexity, and to the current landscape of computational neuroscience surrounding it, we are now in a position to delve into the methodological foundations of this work. The following chapter introduces the modeling frameworks and computational tools employed in the project, which is the core of this thesis.

Chapter 3
The modeling framework: tools, materials and methods

This chapter is dedicated to presenting the modeling tools utilized in the project work (see **Chapter 4**), their features and the methodologies adopted in their usage. We will start by addressing the Brain Scaffold Builder (BSB) tool, developed by the laboratory of rain and Behavioral Sciences of the University of Pavia, to then delve into presenting some specific simulators, offering an example for both multi-compartmental neurons simulations (the NEURON simulator) and point-neuron simulations (the NEST simulator), to then address an extension of NEST, namely NEAT, which makes reduced versions of multi-compartmental neuron models.

3.1 BSB

As anticipated earlier, the Brain Scaffold Builder (BSB) was developed by the laboratory of Brain and Behavioral Sciences of the University of Pavia and it is to be intended as a black box component framework for multi-paradigm neural modelling. It is designed to assist in the modeling and simulation of brain structures at different levels of complexity, as it enables the definition, generation, and manipulation of detailed three-dimensional brain architectures. The tool allows precise spatial distribution, cell types, connectivity rules, and microstructural properties, providing a biologically plausible environment for subsequent reproducible simulations.
The BSB provides a Python interface and it is fully compatible with Python versions 3.9, 3.10 and 3.11 for now.
It uses a precise workflow that revolves around BSB's so called 'components', the modules of the tool. The main ones are: topology, cell types, morphology, placement, connectivity and simulation. This paragraph is entirely dedicated to address them and their main features.
It is completely open-source, meaning that all the code that constitutes it is made freely available for anyone to view, use, modify, and share, in order to promote transparency and cooperative development of the tool itself, which can be found in the following GitHub repository: `https://github.com/dbbs-lab/bsb`.

3.1.1 Topology

The topology module enables abstract modeling of a region's spatial layout by defining shapes (partitions) and organizing them hierarchically into regions. The topology forms a tree structure ending in terminal partitions. Initialization starts with a network size hint

at the root, which subdivides it among children. Once handed back the initial layouts of their children, parent regions can propose transformations to finalize the layout.

3.1.2 Cell types

A cell type represents an abstract description of a cell population, which is placed within partitions based on specified placement rules. Morphologies and orientations can be assigned to cell types. During placement, cell positions are generated as a `PlacementSet`, which can later be organized into `ConnectivitySets`. In simulations, cell types are instantiated as specific cell models. It is also possible to specify spatial densities and morphologies.

3.1.3 Morphology

Morphologies are 3D representations of cells, composed of head-to-tail connected branches, where each branch consists of a series of points with associated radii. Points can carry labels and multiple user-defined properties. The root branch typically resembles a soma due to its larger radii, and can have multiple child branches.
In network configurations, morphologies are defined under the `morphologies` key and managed within the `MorphologyRepository` for assignment to cells. Within the BSB one could also parse and construct morphologies and modify subtrees.

3.1.4 Placement

The placement block in the configuration defines how cells are positioned within partitions. All placement strategies are derived from the `PlacementStrategy` class and must specify how each `CellType` is distributed within a `Partition` volume. BSB provides several built-in placement strategies, while custom strategies can also be implemented. The main placement strategies available in BSB, using kd-tree partitioning of the 3D space are particle placement, parallel array placement and satellite placement. Placement data for each cell type is stored in corresponding `PlacementSets`.

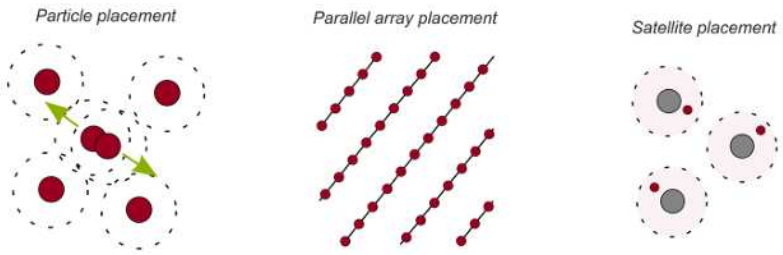


Figure 3.1: Main BSB placement strategies

3.1.5 Connectivity

Connections between cells are managed using `ConnectivityStrategy` objects, which define the rules for establishing connections. BSB offers a set of built-in strategies, while custom strategies can also be developed. The main connection strategies available in BSB are distance-based in/out degree probability functions, voxel (or fiber) intersection based on voxelization of morphologies and touch detection. Once generated, connections are stored in `ConnectivitySets`.

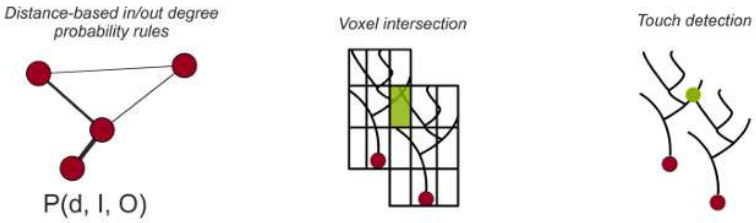


Figure 3.2: Main BSB connection strategies

3.1.6 Simulation

The Brain Scaffold Builder provides adapters that allow network models to be simulated using widely adopted neural simulation software. Once created, models can be executed across different platforms without requiring modifications. Currently, adapters are available for NEST, NEURON, and ARBOR, although ARBOR support is still under development.

As a closing remark, one could also write their own components and amplify the BSB framework [15, 1].

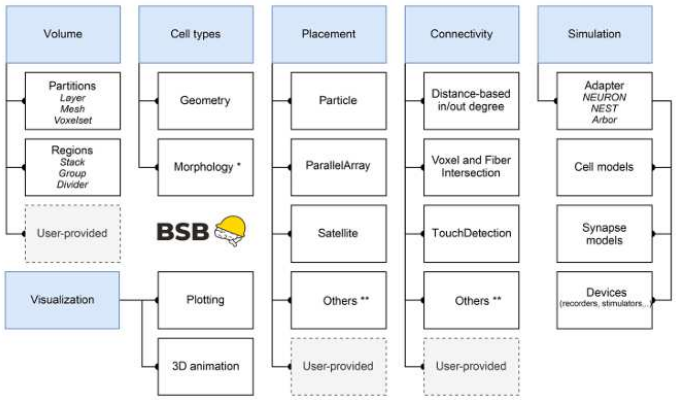


Figure 3.3: Core BSB operations within the mentioned modules

3.2 NEURON

NEURON is a simulation environment designed for modeling individual neurons and networks of neurons. It is particularly well-suited for simulations involving detailed neuronal morphologies and complex electro-physiological properties.
The tool was originally developed at Yale University while today it is maintained as an open-source project with significant support from the Blue Brain Project at EPFL.
NEURON supports the creation of multi-compartmental models, allowing to represent the spatial structure of neurons and to simulate the electrical activity across different parts of the cell. Among its main features, NEURON offers tools for building, visualizing, and analyzing neuronal morphologies, specifying biophysical mechanisms, and running simulations under a variety of conditions, providing a graphical interface as well as scripting capabilities via HOC and Python languages [25]. In the project work, the version 8.2.6 was utilized, and it was interfaced with the BSB.

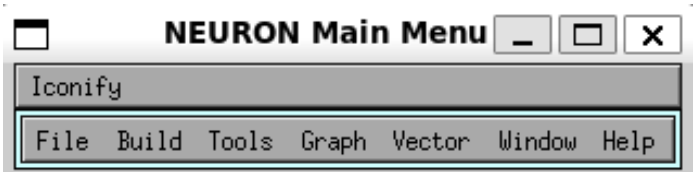


Figure 3.4: The main menu of the NEURON simulator, displaying the main functions of the tool

The following sections provides some insights on the core elements of a multi-compartmental model of a neuron, by analyzing the biological structures these models are based on.

3.2.1 Components of the neurons

To fully understand what it means to model a neuron, or a network of neurons, by adhering to its morphology, it is necessary to briefly address said morphology. This paragraph presents a short description of the main components of neurons.
Let us start by stating a very distinctive property that characterizes neuronal cells: they are excitable. Neurons can in fact receive activation (or inhibition) thanks to electrical signals which they are able to transmit and which carry information that is encoded, elaborated, and decoded as spikes fired by neurons: the action potentials. The signal is first detected by the dendrites of the neuron, which serves the role of an input zone. Spatial and temporal integration of the signal occurs across the dendrites, and, if the threshold that makes the neuron fire is surpassed (which means, the neuron stops being at its resting potential, generally around -70 mV, and reaches its reversal potential), an action potential is generated in the soma of the neuron (the body of the cell that contains the nucleus), and gets propagated from the axon hillock or axon neck down the axon body, helped by the presence of the myelin sheet, a fatty substance that facilitates the conduction by isolating the membrane and by making the conduction saltative, as the electrical signals travels by "jumping" in the gaps created by the sheet, the so called nodes of Ranvier. Since the signal works by depolarizing the cell (since it is much more negative on the inside, that on the outside), depolarizing a smaller region (the node) is much more convenient and faster than depolarizing the whole membrane. Once the signal reaches the axon terminal, it is propagated in the synaptic cleft (the space between two spatially contiguous neurons) in three main possible ways: through chemical synapses, that release chemicals in vesicles called 'neurotransmitters' that are detected by receptors located in the dendrites of the following neuron, through electrical synapses, that leave the signal unaltered and propagate it to the next neuron, or through mixed synapses that combine the previous two [23].

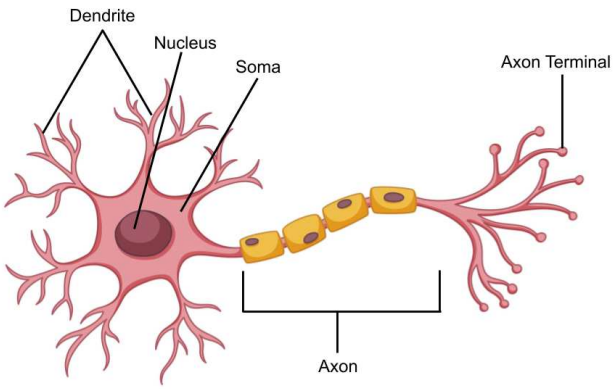


Figure 3.5: A schematic representation of the morphology of a neuron

The **Figure 3.5** represents a neuron with its main components. Notice that this is just an exemplificative morphology: in the brain we can find thousands of different-looking neurons.

3.2.2 The ionic channels

In the previous paragraph we talked about depolarization of the cell membrane, but did not explain the mechanism at the core of this process: the ionic channels.
This section shortly addresses what they are, how they work and why they are relevant for multi-compartmental, or conductance-based, models of neurons.
Ionic channels are proteins composed of multiple sections that lie in the phospholipid bilayer that constitutes the neuronal membrane. Other kinds of proteins located in the membrane are leakage channels and pumps, which are both pores that help maintaining the adequate intra- and extracellular concentration of ions (as mentioned before, negativity inside the cell is preserved to keep the neuron at its equilibrium or resting potential). Ionic channels play a particular role though, as they are responsible for the process of depolarization of the cell: these pores are more permeable to specific ions, the main channels at play being potassium (K+) channels, sodium (Na+) channels, calcium (Ca2+) channels and chloride (Cl-) channels. By making the aforementioned ions flow through them, creating ionic currents, the ionic channels are able to modify the concentration of ions inside and outside of the membrane, changing the membrane potential by raising it to the adequate threshold that makes the neurons fire when the cell is depolarized. After, the cell generally hyper-polarizes or generally re-polarizes back to the starting condition, and is possibly newly excited.
These channels can be classified based on their gating mechanisms, including voltage-gated channels, which open in response to changes in membrane potential, and ligand-gated channels, which are activated by specific chemical signals. Other types include mechanically gated and temperature-sensitive channels [32].

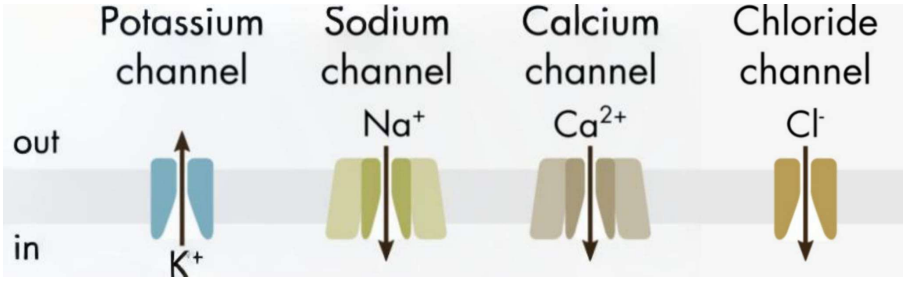


Figure 3.6: A schematic representation of the main ionic channels present on the membrane and the ionic currents

The **Figure 3.6** represents a simplified representation of the main ionic channels and the direction of the currents of the ions through the membrane.
Ionic channels are crucial in multi-compartmental neural modeling as they are one of the main components of these models, that are more faithful to the morphology and feature these kind of pores on their so called compartments. Let's notice that increasing physiological realism implies greater computational effort, reasonably.

3.3 NEST

NEST (NEural Simulation Tool) is a simulation environment specifically designed for large-scale modeling of spiking neural network dynamics. It is optimized for handling networks composed of many thousands to millions of neurons and billions of synapses,

making it ideal for investigating the collective behavior of complex neuronal systems.
The development of NEST is led by the NEST Initiative, with significant contributions from the Blue Brain Project at EPFL, among others. NEST is highly parallelized and can run efficiently on a wide range of hardware, from laptops to supercomputers.
Unlike simulators focused on detailed single-neuron morphology and biophysics, NEST emphasizes efficient and scalable simulation of point neurons, such as integrate-and-fire or Hodgkin-Huxley type models (see section 3.3.1). It is mostly adopted to represent neuron in a point-like manner, as if they were composed of a single component, ideally their soma. This reduction makes computation way more efficient and extensive, but there is a reasonable loss in biological realism and one could not investigate single ionic channel activity and other local properties, but only spiking activity from network to at most single cell level.
With the tool, one could manually create neurons as "nodes", set their parameters, connect them into a network and simulate their activity, although NEST also provides a rich library of neuron, synapse, and stimulation models, and users can easily configure network structures and simulation parameters through a high-level Python interface (PyNEST) [64, 20].
In the current work NEST version 3.7.0 was utilized (more information in section 4.2) and it was interfaced with the BSB as well.

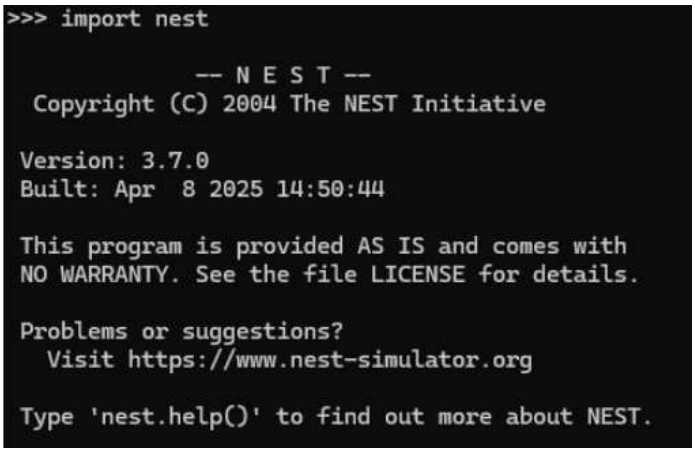


Figure 3.7: The NEST interface opened from the Python Shell

3.3.1 Point-neurons computational modeling

Some examples of computational models for point-neurons were provided in the previous paragraph: the integrate-and-fire model and the Hodgkin-Huxley model. This paragraph addresses them briefly.

The integrate-and-fire model is a simplified point-neuron model that captures the essential behavior of neurons: it integrates incoming synaptic inputs over time, generally according to a simple differential equation, typically governed by passive leakage and synaptic input, and when the membrane potential Vm reaches a certain threshold Vth, it generates a spike and resets to its baseline value. The equation looks like this:

$$c_m \frac{dV}{dt} = -\bar{g}_L(V - E_L) + \frac{I_e}{A}$$

In this context c_m is the specific membrane capacitance, $\bar{g}_L$ is the specific leak conductance, E_L is the leak reversal potential, I_e represents the external injected current, and A is the membrane area. The model is computationally efficient and widely used in large-scale network simulations [19].

On the other hand, the Hodgkin-Huxley model provides a biophysically detailed description of neuronal activity. It models the membrane potential based on the dynamics of ionic channels (e.g., sodium and potassium channels) through a set of nonlinear differential equations. While much more computationally intensive, the Hodgkin-Huxley model offers a more accurate representation of the electrophysiological behavior of real neurons. The set of equations looks like this:

$$\begin{cases} i_m = \bar{g}_L(V - E_L) + \bar{g}_K n^4(V - E_K) + \bar{g}_{Na} m^3 h(V - E_{Na}) \\ c_m \frac{dV}{dt} = -i_m + \frac{I_e}{A} \\ \tau_n(V) \frac{dn}{dt} = n_\infty(V) - n \end{cases}$$

Here i_m is the total membrane ionic current density, $\bar{g}_L, \bar{g}_K, \bar{g}_{Na}$ are the maximum conductances for the leak, potassium, and sodium channels respectively, E_L, E_K, E_{Na} are the corresponding reversal potentials, and I_e is the externally applied current. The function $\tau_n(V)$ defines the voltage-dependent time constant for the gating variable n, determining the speed at which n approaches its steady-state value $n_\infty(V)$. The gating variables, such as n, represent the probability of a subunit gate being in the open state and evolve according to voltage-dependent dynamics [26].

3.3.2 NEAT

NEAT (NEural Analysis Toolkit) is a Python library, to be intended as an extension of the NEST ecosystem and developed by Willem Wybo, postdoctoral researcher and group leader of the Dendritic Learning Group (DLG) at EFPL.
This library is designed for the study, simulation, and simplification of morphological neuron models. It supports neuron morphologies in the standard `.swc` format and provides various tools for interacting with and analyzing neuronal structures [69].
A key feature of NEAT is its advanced algorithm for the simplification of complex neuron morphologies into compact compartmental models, maintaining simulation accuracy while greatly reducing complexity. A schematic representation of a neural model reduction is displayed in the **Figure 3.8**: after selecting some locations we wish to keep, the neural structure gets reduced. These simplified models can be interfaced with NEURON and, since a very recent update, also with NEST [71].

Structurally, NEAT is based on a hierarchy of tree classes, beginning with `neat.STree` and extending to specialized structures like `MorphTree` and `PhysTree`, which support morphological and biophysical data handling, respectively. More specifically, the `MorphTree` class allows you to navigate the morphology of neurons (to be loaded as an `.swc` file) by structuring their compartments via a set of nodes accessible via indexing, while the `PhysTree` class allows the definition of physiological and electrical parameters across the tree, especially by specifying the ionic channels, their location and their parameters in the model. In this framework, these channels are implemented via the Hodgkin-Huxley paradigm, previously presented. This class also works by providing a starting `.swc` file for the specification of the neural structure. Additional modules provide tools for other purposes, such as the class `NeuronSimTree` which takes a physical tree as input and allows to simulate its activity. On the other hand the classes `CompartmentFitter` and `NeuronCompartmentTree` help automatically reducing a physical tree into a few-compartmental model and simulating it, respectively. Finally, additional modules provide advanced computational tools such as `GreensTree`, used to compute impedance matrices across neurons and `NestCompartmentTree` which exports reduced models into NEST and allows to handle and simulate them using the syntax of NEST instead of NEAT's, at single-neuron level [70].
The version of NEAT utilized in the project work is 1.0.0 and a more in-depth exploration of its characteristics is provided in **Section 4.5**.

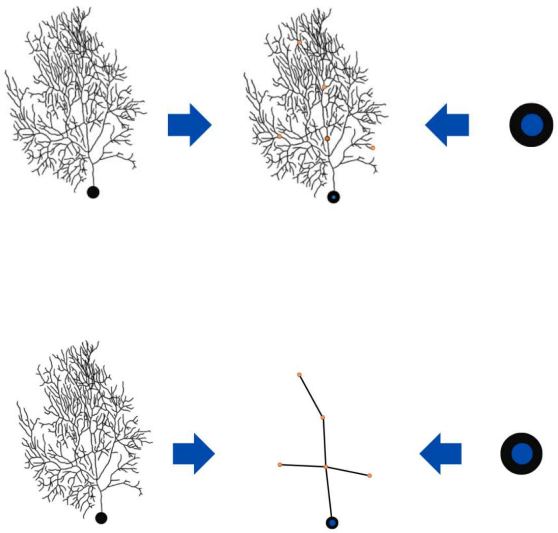


Figure 3.8: A schematic representation of the simplification process in a Purkinje cell

Other simplification tools

Although not part of the NEST ecosystem, some other simplification tools aside NEAT have been developed. In this section, we specifically focus on the Rössert-Pozzorini pipeline and AdEx models.

The Rössert-Pozzorini pipeline for multi-compartmental networks simplification provides point-neuron networks as output, and is mainly composed of two steps, called soma-synaptic correction and neuron simplification. While the **Figure 3.9** shows a schematic representation of the pipeline, the details of this pipeline are the following: in the first step dendritic synapses are moved to the soma and some low-pass filters are applied to synaptic currents. Furthermore some parameters for excitatory and inhibitory activity get calculated and eventually clusetered. At this stage three types of networks get produced: the original, multi-compartmental one (in blue in the figure), one where the synapses are moved to the soma (in orange in the figure) and one where both the synapses are moved to the soma and the aforementioned corrections with filters and clustering are applied (in green in the figure).

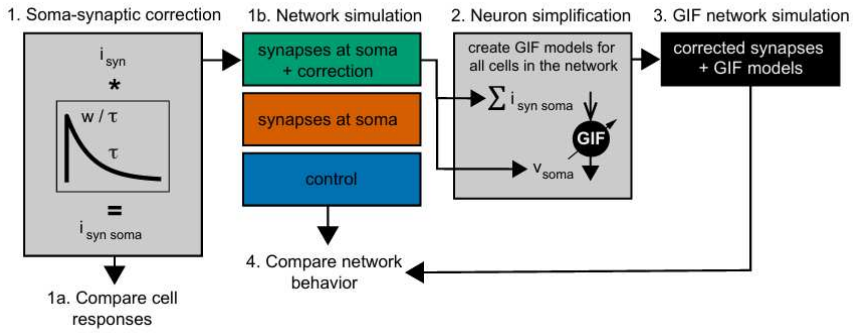


Figure 3.9: A schematic representation of the Rössert-Pozzorini pipeline

The **Figure 3.10** represents neurons in each of these networks, which are followingly simulated. For the second step, the neurons of the corrected network get simplified through a mathematical process developed by Pozzorini which is based on averaging functions and on the summation of synaptic currents [53]. After this step we obtain a reduced, point-neuron network which we simulate and compare the activity of with the original "blue" network, in order to check for consistency. The work of these authors shows how the "green" and "orange" networks reproduce the activity of the original one faithfully, the "green" one being more accurate, reasonably [56].

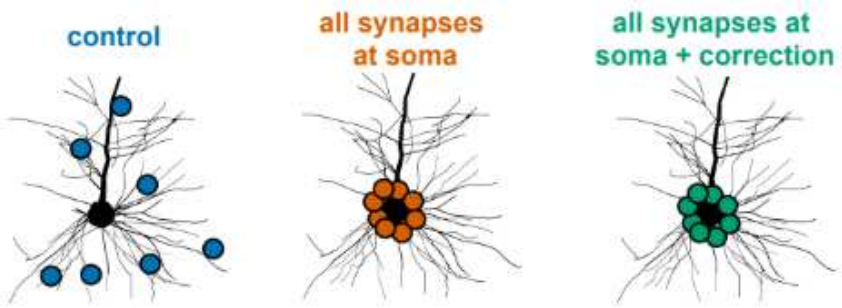


Figure 3.10: A schematic representation of the Rössert-Pozzorini pipeline

AdEx models, on the other hand, are an alternative simplification tool which reduces multi-compartmental models into single-point neuron models as well. They can be seen as an alternative to Pozzorini's integrate-and-fire simplification process. These models account for two specific features which are lacking in classical (point-neuron) integrate-and-fire models: exponential spike initiation, which captures the rapid upstroke of the action potential and an adaptation current (w) which models spike-frequency adaptation, as neurons get less responsive over time, and resonance to specific input patterns. AdEx models are defined by the following equations:

$$\begin{cases} C\dfrac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T \exp\left(\dfrac{V - V_T}{\Delta_T}\right) + I - w \\ \\ \tau_w \dfrac{dw}{dt} = a(V - E_L) - w \end{cases}$$

The first equation governs the evolution of the membrane potential V over time. It includes a leak current term, proportional to the difference between the membrane potential and the leak reversal potential E_L, and an exponential term that models the exponentially rapid upstroke of the action potentials. The term I represents the input current, while w is an adaptation variable that introduces a negative feedback effect on the membrane potential, contributing to spike-frequency adaptation. The second equation describes the temporal evolution of this adaptation variable. It depends on the difference between the membrane potential and the leak potential, scaled by a parameter a that determines the strength of the adaptation. The time constant τ_w sets the timescale over which this adaptation builds up and decays.
Together, these equations allow the AdEx model to reproduce a wide variety of firing behaviors, ranging from regular spiking to bursting, by tuning a relatively small number of biologically interpretable parameters [49].

While these two tools will not be explored in the current document, but can be seen as an addition to explore the world of automatic simplification tools, the next chapter is dedicated to reporting a project work developed during this academic year which features all the other tools that were mentioned in this chapter, namely NEURON, the BSB, NEST and NEAT.

Chapter 4
Project work

4.1 Introduction

This chapter contains the core of this thesis: the project work that was developed during the current academic year.
Each of the sections of this chapter is dedicated to one of the main blocks, we could refer to them as steps, that constituted the whole process which led to the completion of the project. They are mainly divided based on the tools used and the finality of each of them. For all of them, aim behind their developement, the process of developement itself and discussion of the results provided is reported.

The focus of the project work was to explore different kinds of neural models, specifically of the Purkinje cell, which, let us recall, is located in the cerebellar cortex, and to carry out simulations featuring said neuron, by starting with a multi-compartmental approach within the NEURON simulator framework, to then delve into a whole-network point-neuron cerebellar circuit reconstruction and simulation within the BSB framework interfaced with the NEST simulator, to ultimately address a reduced, few-compartmental model of the aforementioned cell, obtained with the tool NEAT, and to investigate its activity such as in the previous steps.
Let us now show each of these steps in a systematic and comprehensive way, to ensure reproducibility.

4.2 The set up

4.2.1 Aim

During the developement of the project I was working on a HP Pavilion laptop 15-eg1xxx PC, based on x64, having Windows 11 as its operating system, and with Python installed in the version 3.12.
The first objective of the project was to create a functional Python virtual environment that contained all the needed tools and could work from the Linux terminal, as NEST (and therefore NEAT) and NEURON are not directly compatible with Windows, and with a Python version contained in the range 3.9-3.11, as the BSB is only fully compatible with those versions.

4.2.2 Development

The set up was prepared in the following way: as an initial step the command prompt was opened and the following line was entered.

```
C:\Users\Anna Clara Arrigoni>wsl --install
Per eseguire l'operazione richiesta è necessaria l'esecuzione con privilegi elevati.
Installazione in corso: Piattaforma macchina virtuale
Piattaforma macchina virtuale è stato installato.
Installazione in corso: Sottosistema Windows per Linux
Sottosistema Windows per Linux è stato installato.
Installazione in corso: Ubuntu
Ubuntu è stato installato.
L'operazione richiesta è stata eseguita. Le modifiche avranno effetto al riavvio del sistema.
```

The Linux subsystem for Windows was installed, and so was Ubuntu, a free and open-source operating system based on the Linux kernel. To install the latest Ubuntu version (24.04.1 LTS), you can either download it directly from the Microsoft Store or install it via the command line.
First, list the available distributions by running:

```
wsl --list --online
```

Then, install Ubuntu 24.04 with:

```
wsl --install -d Ubuntu-24.04
```

After installation, launch the Ubuntu executable (`ubuntu2404.exe`) to initialize the subsystem, or simply search for "WSL" in the Start menu. Notice that a username and a password will be required the first time. Once Ubuntu is installed, I updated the package lists and upgraded the system to the latest versions of installed packages by running:

```
sudo apt update
sudo apt full-upgrade -y
```

The first command refreshes the list of available updates, while the second installs all available upgrades without prompting for confirmation.

When the Linux subsystem was set, it was time to deal with the installation of the right version of Python. I opted for Python 3.10 to have the same environment used in the laboratory in the department of Brain and Behavioral Sciences of the University of Pavia. This version is not directly compatible with Ubuntu, therefore I cloned the repository `deadsnakes`, which contains old Python versions, and installed Python 3.10, along with the virtual environment and development packages. Here I report the commands I launched and the sanity check for the installation.

```
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt install python3.10 python3.10-venv python3.10-dev -y
python3 --version
```

The output of this last command was: `Python 3.10.16`.

What is left is to address is the creation of the virtual environment and the installation of all the tools mentioned in **Chapter 3**. For the sake of making a complete report of the process, at the very beginning I had created a folder on my Desktop in order to put my virtual environment inside of it and work directly from there, but it did not end up being a successful decision because, as it will be explained in a moment, I ended up installing a version of NEST (NEST 3.7) provided by the laboratory in the department of Brain and Behavioral Sciences of the University of Pavia, which I could only unzip and extract from the C disk of my PC, which is the reason why I then created a folder directly inside of my C disk and re-created the virtual environment with all the needed installations there.
While in the following part I will address the smooth process of setting up the environment in the right folder, I also report in this paragraph that initially I did not install NEST in the environment, as I had a pre-installed version of it on my PC coming from some

laboratories I took during the first semester, that was exclusively compatible with Python 3.12, which I recall was substituted with the version 3.10. Hence I uninstalled NEST and re-installed it following the installation instruction from the NEST website [69]. The installed version of NEST ended up being NEST 3.8, which followingly showed incompatibility with the BSB, and which was uninstalled as well. Ultimately, as mentioned previously, the right version, NEST 3.7, compatible with both Python 3.10 and the BSB was installed. In the next paragraph the complete process is addressed.

As mentioned previously, I created a folder in the C disk of my computer and called it 'Anna'. Then, I accessed it from the WSL terminal using the `cd` command until I reached it (the full command path was: `cd /mnt/c/Anna`) and I created a Python virtual environment, where the work would have been carried out, with the following command:

```
python3 -m venv env_bsb
```

Then, I activated the virtual environment:

```
source env_bsb/bin/activate
```

Once inside the environment, I proceeded to install the `bsb` package:

```
pip install bsb
```

Finally, I performed a basic sanity check by running the Python interpreter and importing the `bsb` module:

```
python
>>> import bsb
```

After that, the simulators were installed from the WSL command prompt, first via their versions interfaced with the BSB via the following commands:

```
pip install bsb[neuron]
pip install bsb[nest]
```

and later in their completeness.

To install NEURON, running the following command was enough:

```
pip install neuron
```

Then, one could check the correct installation by launching the following lines from the terminal:

```
which nrngui
nrngui -python
```

The first command checks if the `nrngui` executable is in the system's path and returns its location if installed correctly. In my case it returned `/mnt/c/Anna/env_bsb/bin/nrngui`. The second command launches the NEURON graphical user interface with the Python interface, enabling Python-based interaction with the simulation environment if the installation was successful.

As anticipated earlier, installing NEST resulted in being slightly more complex. Firstly I needed to unzip the installation package, previously loaded in the folder 'Anna', directly in there, and from the WLS terminal. The command I used was the following:

```
xtar -xvf nest-simulator-3.7.tar.gz
```

This command uses `xtar`, a tool for extracting files from compressed archive formats. The flags used are:

- `-x`: Extract the contents of the archive.
- `-v`: Verbose mode, which provides details on the extraction process.
- `-f`: Specifies the archive file to extract, in this case `nest-simulator-3.7.tar.gz`.

After this step, with the virtual environment being active, I created a new folder inside the 'nest-simulator-3.7' one (after accessing the latter through the command: `cd nest-simulator-3.7`), called 'nest-build' with the following command:

```
mkdir nest-build
```

I then entered the directory with the command `cd nest-build`, and proceeded to launch:

```
cmake .. -Dwith-mpi=ON
```

This command configures the installation files for compilation. The options used are:

- `..`: Specifies the parent directory as the location of the source code.
- `-Dwith-mpi=ON`: Enables the use of MPI (Message Passing Interface) for parallel computing during the build process.

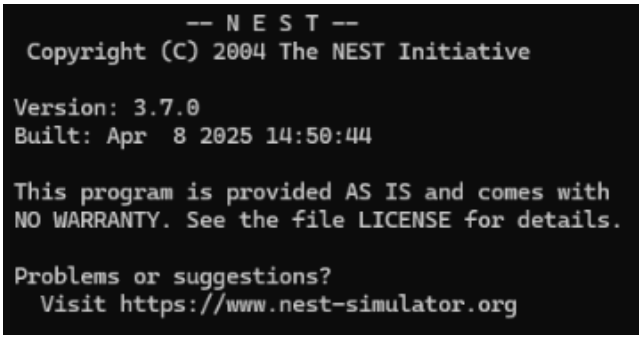
After that some final commands were launched:

```
lscpu
make -j4
```

The first command displays detailed information about the CPU architecture of the system, such as the number of cores, CPU model, and other hardware specifications.
The second command compiles the installation files using the `make` build system. The `-j4` flag specifies that the build process should use 4 cores for parallel compilation, because in my case there was four of them as the previous command showed me, speeding up the process by running multiple tasks simultaneously.
To conclude the installation I ran the command `make install` and did a sanity check with Python in the following way:

```
python
>>> import nest
```

Alternatively, one could simply call `nest` from the terminal. This should appear in either case if the installation was successful:



The last tool to be installed was NEAT. To install it I followed the directions offered on their github [69] while operating in the 'Anna' folder.

```
git clone https://github.com/unibe-cns/NEAT.git
cd NEAT
pip install .
neatmodels install default
neatmodels install default -s nest
python exe -m pip install crlibm
```

The commands perform the following actions:

- `git clone https://github.com/unibe-cns/NEAT.git`: Clones the NEAT repository from GitHub to the local machine.
- `cd NEAT`: Changes the directory to the newly cloned NEAT repository.
- `pip install .`: Installs the NEAT package and its dependencies using `pip`.
- `neatmodels install default`: Installs the default models required by NEAT.
- `neatmodels install default -s nest`: Installs models specific to the NEST simulator.
- `python -m pip install crlibm`: Installs the `crlibm` library, which is required for certain numerical operations.

Also in this case we can verify the correct installation by running a sanity check with Python in the following way:

```
python
>>> import neat
```

The same NEST page as before should appear if the installation was succesful.

4.2.3 Results and discussion

All the previous process led to the creation of a Python virtual environment, running on Python 3.10 and accessible from the Linux Subsystem for Windows, that contained all the tools needed for the project. While this section was an overview of the setup of the environment I worked within, the next sections will address the core steps of the project work one by one.

4.3 Multi-compartmental morphologies visualization, simulations and data analysis in Purkinje cells

4.3.1 Aim

This section of the project work was aimed at familiarizing with the NEURON tool, and to visualize the morphological structure (on the basis of `.swc` files) or simulate the activity (or do both operations) of different models of the Purkinje cell.
For the sake of completeness, morphological visualization was also tested on a pyramidal cell, taken from the GitHub repository of NEAT, included in the documentation of the tool [69], and the simulation of the activity of a stellate cell was also investigated [55]. However, in this paragraph, only the process and results involving Purkinje cells are reported, in order to have some degree of consistency with the very topic of this thesis.

4.3.2 Development

Morphologies visualization

In order to better understand how multi-compartmental models work, it is indeed adequate to look into the morphology of the neural cells they simulate the activity of.
In this section, the visualization process and graphical results of said morphologies is reported, starting from two different `.swc` files representing neural morphology of Purkinje cells via 3D coordinates: one coming from the GitHub repository of NEAT [50] (created by Miyasho Tsugumichi) in the section `examples/models/morphologies`, the other coming from the Github repository of the laboratory of Brain and Behavioral Sciences of the

University of Pavia [14], located in the section `cerebellum/morphologies`. I stress that this repository is private, at the moment, as it will be addressed in **Section 4.2.2**. Some other morphologies is `.swc` format can be found on NeuroMorpho.Org, an online inventory of digitally reconstructed neurons and glial cells (support cells of the central nervous system) associated with peer-reviewed publications [51].
To start this process, it is needed to locate yourself in the main folder ('Anna', in my case) where the project is being developed, with the virtual environment being active. From here, run the following commands:

```
cd bsb_env/bin
nrngui
```

The first command, `cd bsb_env/bin`, navigates to the `bin` directory of the `bsb_env` environment, which contains the NEURON executables, while the second command, `nrngui`, launches the NEURON graphical user interface, allowing interaction with the simulation environment.
Once opened, the main menu of the graphical interface should look like the one in **Figure 3.4**.
From here it is enough to click the buttons `Tools>Miscellaneous>Import 3D`, and navigate directories to load the `.swc` file you need. In my case, since I launched the command while being in the 'bin' directory, the command to pass to the graphical interface was: `../../NEAT/examples/models/morphologies`, selecting the file `purkinje1.swc` to open the Purkinje cell morphology created by Miyasho.

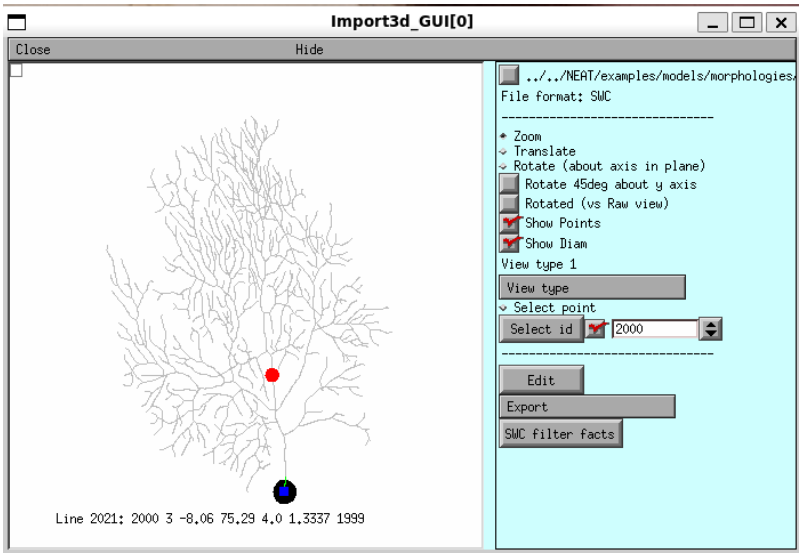


Figure 4.1: Miyasho's morphology of the Purkinje cell

The **Figure 4.1** reports how the morphology of Miyasho's Purkinje cell appeared. Notice that `View type 1` was selected as the modality of visualization, so that each branch could be clearly distinguishable, and so that the soma (in blue) and its diameter could be visible. Furthermore a random point on the dendrites was selected for the sake of making an example (in red).
The same process was repeated to visualize the Purkinje cell morphology developed in the laboratory of Brain and Behavioral Sciences of the University of Pavia. Of course, however, it was necessary to substitute the previously inserted directory, the one passed to the interface to load the desired `.swc` file, with the following one:
`../../cerebellum/morphologies` (see the process of cloning of this directory in **Section 4.4.2**), selecting the file `PurkinjeCell.swc` in this case, and adopting analogous visualization modalities, to ensure some degree of comparability, as it will be addressed in **Section**

4.3.3.
The **Figure 4.2** reports how the morphology of the Purkinje cell developed by the laboratory of Brain and Behavioral Sciences of the University of Pavia appeared.

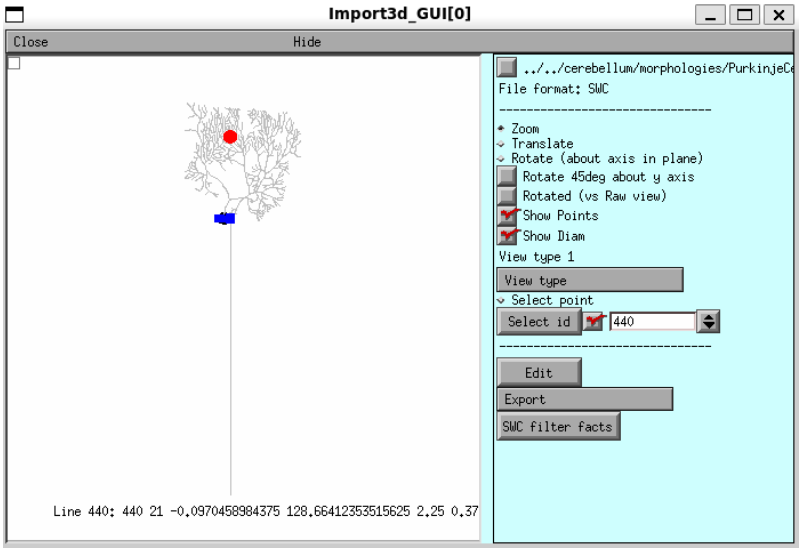


Figure 4.2: Morphology of the Purkinje cell Developed by the laboratory of Brain and Behavioral Sciences of the University of Pavia

Simulations

In this section the simulation under different conditions of Purkinje cells activity is reported. The models that will be shown, both developed in the department of Brain and Behavioral Sciences at the University of Pavia, were downloaded from ModelDB, a database of neural models [46].

The first model is a detailed multi-compartmental model of, of course, Purkinje cells incorporating numerous ionic channels to replicate their complex firing behavior. It was developed to show that, while dendritic calcium channels contribute to bursting activity, the axon, particularly the initial segment and Ranvier nodes, plays a crucial role in pacemaking and spike modulation through sodium-dependent mechanisms. The model reveals how specific ionic distributions and compartmentalization enable Purkinje cells to switch between simple spikes, complex bursts, and bistability, ultimately shaping signal transmission to the deep cerebellar nuclei [43].
The second model compares human and mouse Purkinje cells using morphological reconstructions and ex vivo electrophysiology. While intrinsic firing properties were similar, human Purkinje cells exhibited significantly larger and more complex dendritic trees, with 7.5 times more spines and 6.5 times greater input-processing capacity. These structural differences (above all, dendritic complaxity is analyzed) suggest an evolutionary enhancement of computational power in human cerebellar circuits [42]. Although code for simulation was available for the human Purkinje cell as well, in this document only the simulation of the activity of the mouse Purkinje is reported, to keep the work as consistent as possible.

Let us start by documenting the simulation of the first model. Two conditions were investigated: spontaneous firing, it is important to recall that Purkinje cells are autorhythmic and can produce some degree of activity independently, and activity under injection of a positive electrical current. The first thing to do was to locate myself in the correct directory. While the virtual environment was still active, I located myself in the designated

folder on my Desktop, as I had downloaded the models directly on it, each in a separate folder. For me, the directory from the WSL command prompt looked like this:

```
mnt/c/Users/Anna Clara Arrigoni/Desktop/masoli2015/purkinjecell_2015
```

The first command to run was:

```
nrnivmodl mod_files/
```

The files present in the folder `mod_files` contain the necessary parameters and information regarding the ionic channels that define the cell's electrochemical properties. The `.mod` files were compiled using the NEURON tool `nrnivmodl`, which generates the corresponding compiled object files needed for accurate simulation of ionic channel dynamics.

After compilation, one could start investigating specific simulation conditions by calling `.py` files contained in the folder `protocols`, which contain code that sets up the simulation conditions for NEURON to display it in a second moment. The protocol files contain the following line: `from Purkinje_py3 import Purkinje_py3`, which loads a model of the Purkinje cell described in the file `Putkinje_py3.py`. For both the protocols tested I set up the simulation time to be 2000 ms. The code line containing this specific information is the one reporting `h.tstop = 2000`, where `h` is NEURON's interface to HOC or Python-based simulation control. To run the simulations I passed the following commands to the prompt line:

```
nrngui protocols/02_spontaneous_fire_py3.py
```

for the autorhythmic activity and:

```
nrngui protocols/03_positive_current_inj_py3.py
```

for activity under injection of a positive electrical current.

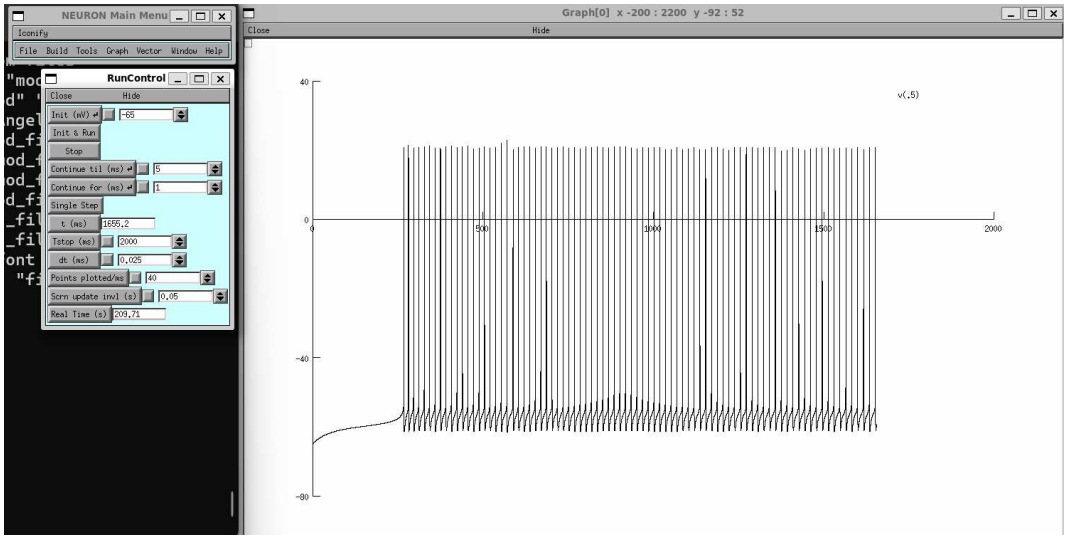


Figure 4.3: Panels displayed by the NEURON graphical interface once called for simulation

The **Figure 4.3** shows what will be displayed by the NEURON simulator after calling for the simulation command in the example of the spontaneous firing. On the left, the main menu is visible, along with the RunControl panel, which contains the commands to handle and visualize the activity of the cell shown in the figure of the simulation, which is on the right. In the picture, I selected from the indicator 'Tstop (ms)' to show all the 2000 ms of the

simulation. While the second protocol tested follows analogously, better graphs and comment on the simulated activity for both protocols will be reported in **Section4.3.3**. Notice that, once the simulations were over, two `.txt` files were automatically saved in the current directory for each simulation protocol (namely, `02_vm_soma`, `02_vm_NOR3`, `03_vm_soma`, `03_vm_NOR3`). They contain tabular data which associates to each time point to a value in the firing activity (membrane potential in the soma for the 'soma' files and in the nodes of Ranvier for the 'NOR3' files). These files were used to produce the aforementioned plots.

Let us now report the process of simulating the second model of the Purkinje cell. In this case, the cell still enjoyed being simulated under the previously mentioned conditions: spontaneous firing and activity under injection of a positive electrical current.
Also in this case, all the necessary material for conducting a simulation was saved in a specific folder on my Desktop, and I started by locating myself in said directory. For me, the directory from the WSL command prompt looked like this:

```
mnt/c/Users/Anna Clara Arrigoni/Desktop/masoli2015/Purkinje_model_human_mice
_2023/mouse_model
```

Then, we can once again compile the `.mod` files and proceed with the simulation of the aforementioned pattern by calling the commands:

```
nrngui protocols/01_Spontaneous_firing.py
```

for the autorhythmic activity, and

```
nrngui protocols/02_positive_currents.py
```

for activity under injection of a positive electrical current.

Once again, the graphical interface of NEURON, opened by the previous commands, looked like the one displayed in **Figure 4.3**.
The `.py` files, contained in the folder `protocols`, also in this case contain code that sets up the simulations. In this case, they import data from the file `Purkinje_morpho_1`, which acesses the `morphology` folder, containing the `soma_10c.asc` file which collects coordinates of the cell and its connections. The file `Purkinje_morpho_1` takes the morphological structure described in the previous file and creates in each compartment of the neuron many ionic channels (specified in the `.mod` files) with respect to their specific parameters.

As two closing remarks, firstly notice that to interrupt a simulation it is enough to press `ctrl+D` and secondly, that also in this case I set up the simulation time in all the `.py` files to be 2000 ms. The line of the code to be modified is the one highlited in **Figure 4.4**.

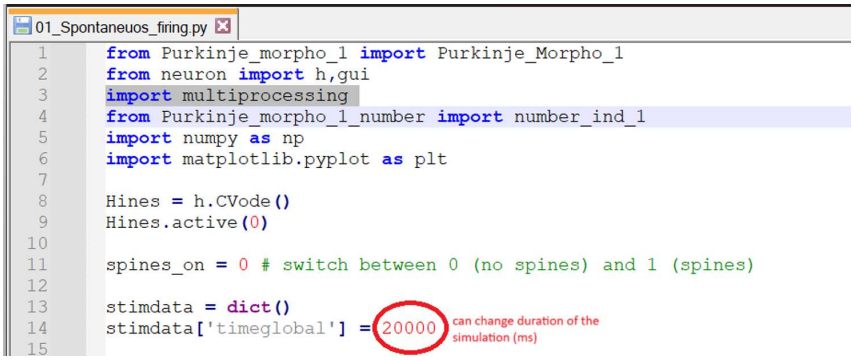


Figure 4.4: Example of code inside the `.py` files

From the **Figure 4.4** one can also notice the line `spines_on = 0`, which sets the presence or absence of spines on the model, which can alter the simulation. Dendritic spines are small

membranous protrusions found on the dendrites of neurons. They serve as the primary sites of excitatory synaptic input and play a key role in synaptic strength, plasticity, and signal integration.

Ultimately, also in this case some output files were stored in the current directory. In my case, for the kind of simulations I ran, they were called `01_vm_soma_no_spines` and `02_soma_no_spines` respectively. Spines were not added in any simulation to be consistent with the first model of the cell. Once again, these files contain the data used to visualize the simulated activity, commented in **Section 4.3.3**.

4.3.3 Results and discussion

Morphologies visualization

By looking at the images of the two reconstructed morphologies we can immediatly notice how Miyasho's Purkinje cell enjoyes a wider, more complex dendritic tree, but lacks in the representation of the axon, while the Purkinje cell developed by the laboratory of Brain and Behavioral Sciences of the University of Pavia has a very long axon and a smaller (but still detailed) dendritic area. Although different from one another, these models are representative of many Purkinje cells we can find in the cerebellum, and once again underline how many different shapes neurons belonging to the same population can have. Ultimately, one can appreciate how the Purkinje cells are one of the biggest cerebellar cells, with an esquisite and detailed morphology, which suggests their relevance in the circuit.

Simulations and data analysis

In this section graphical representations of the simulated activity are presented and commented. To obtain the following plots, working within my setup, it was necessary to open the output `.txt` files, mentioned in the previous paragraph (which contain `csv` data), in LibreOffice Calc. They were loaded in the Italian language. After importation it was needed to save the data as an Excel 2007-365 file in the same working directory of MATLAB, for simplicity in the next step. Once done, to finish the procedure in MATLAB, it was necessary to open the Excel file and to run this simple code each time:

```
% read the data
data = readmatrix('name_of_file.xlsx');
% extract the data
t = data(:,1);   % first column = time
vm = data(:,2);  % second column = Vm
% plot the data
plot(t, vm, 'color_flag');
xlabel('Time (ms)');
ylabel('Vm (mV)');
title('Adequate title');
grid on;
```

Alternatively, one could have imported the output `.txt` files in LibreOffice Calc in the English (USA) language and plotted the results directly in there, by highlighting the two columns, clicking on `Insert graph` and selecting the type of graph to be the linear one, with lines only.
Although both procedures are equivalent in terms of the preoduced result, my choice was uniquely based on aesthetical preference for the clearness of the graph.

Let us start by analyzing the activity of the first model of the Purkinje cell, in the case of spontaneous firing.

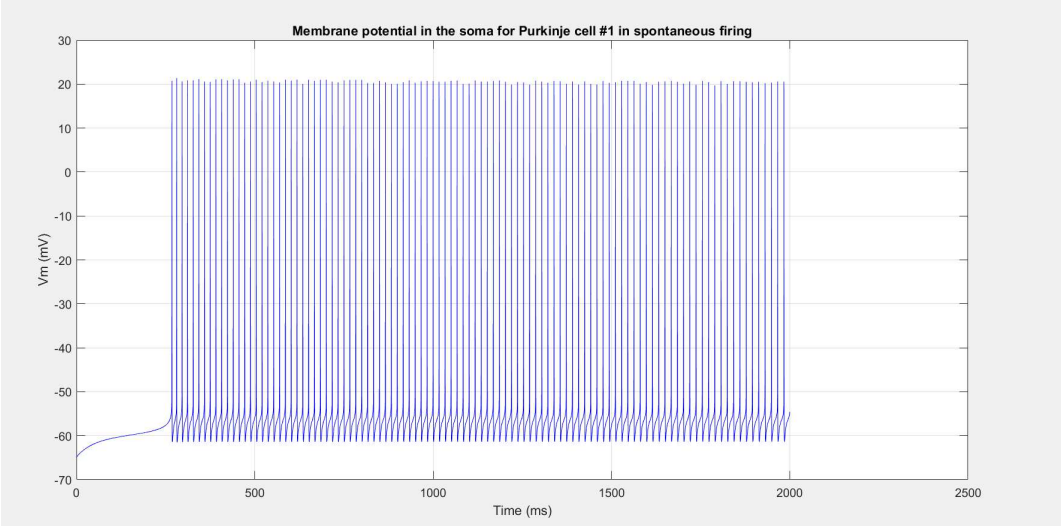


Figure 4.5: Plot of the membrane potential of the soma over time for the first model of the Purkinje cell in the spontaneous firing condition

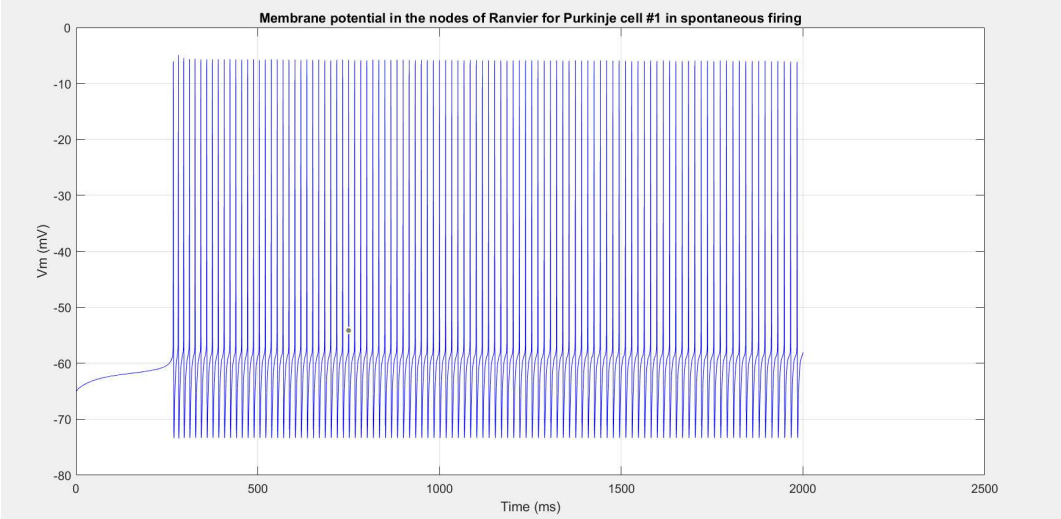


Figure 4.6: Plot of the membrane potential of the nodes of Ranvier over time for the first model of the Purkinje cell in the spontaneous firing condition

By looking at the **Figure 4.5** and the **Figure 4.6** we can notice how in both cases (considering the activity in the soma or in the nodes of Ranvier) the membrane potential, after a brief period of around 250 ms, starts producing a regular activity, with spikes distanced by inter-spike intervals of constant duration, although the depolarization appears to be stronger in the soma.

On the other hand, if we consider the condition of positive current being injected into the cell, we must look at the **Figure 4.7** and the **Figure 4.8** to draw our conclusions.
In both cases, once again, the activity seems to be quite consistent: regular patterns of spiking follow an initial integration period of around 250 ms. However, in this case, both graphs report an even more frequent spiking activity after around 1250 ms of the simulation, as the cell membrane is facilitated in being depolarized thanks to the positive current stimulation it receives, which was absent in the first case addressed. Once again depolar-

ization appears to be stronger in the soma, though.

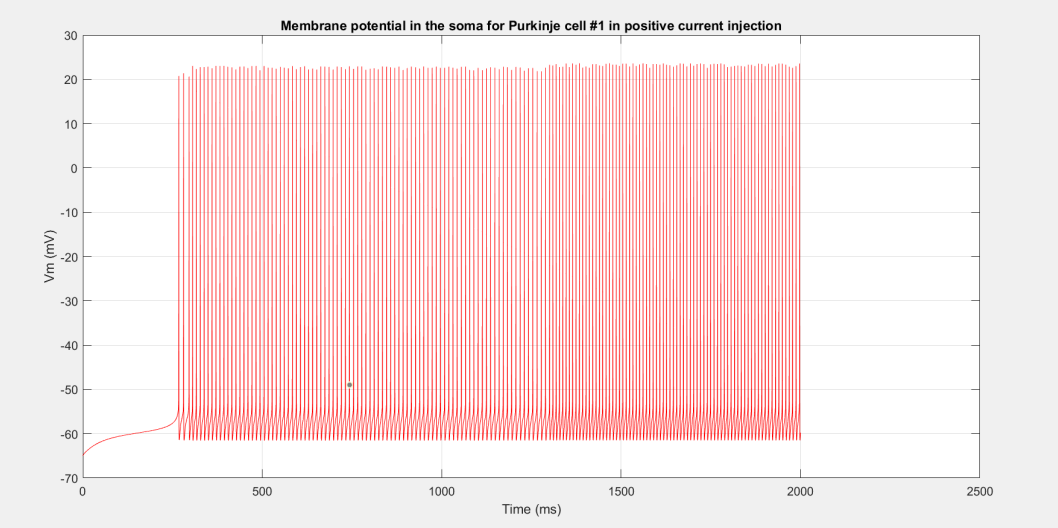


Figure 4.7: Plot of the membrane potential of soma over time for the first model of the Purkinje cell in the positive current injection condition

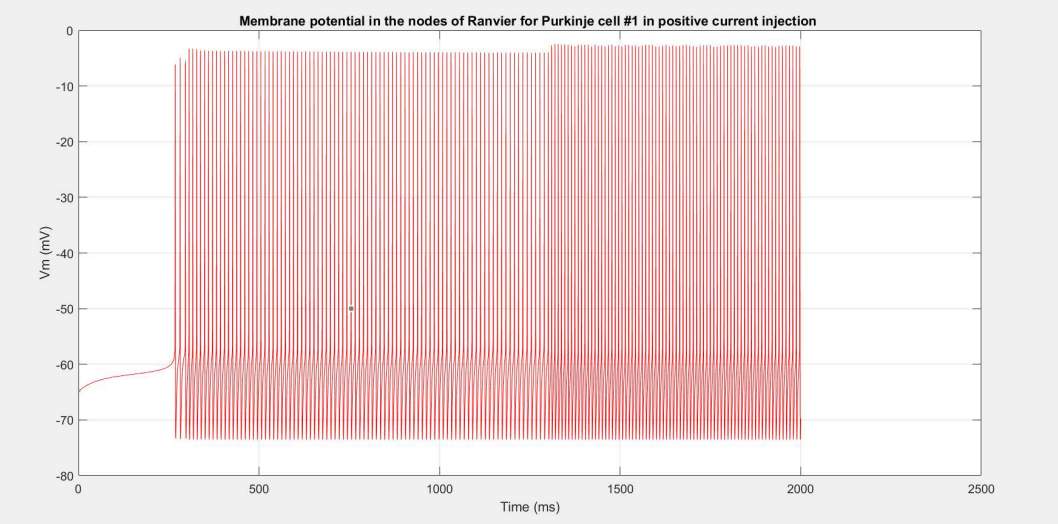


Figure 4.8: Plot of the membrane potential of the nodes of Ranvier over time for the first model of the Purkinje cell in the positive current injection condition

As for the first model of the Purkinje cell both membrane potential in the soma and in the nodes of Ranvier have been investigated. However, the activity only in the soma of the second model of the Purkinje cell will be addressed now, and compared to the previous one. For the sake of comparability, same simulation scenarios correspond to the same color of the line in the plots.

By looking at the **Figure 4.9** we can notice how also in the simulation of this second model of the Purkinje cell, in the spontaneous firing condition, the activity patter remains similiar to the first one, while the depolarization appears to be stronger and the integration period appears to be shorter. The regularity in spiking and the consistency in inter-spike intervals seem to be preserved, although we can notice how the spiking activity appears

to be slightly more frequent in the first case, with longer intervals between spikes in the case under analysis.

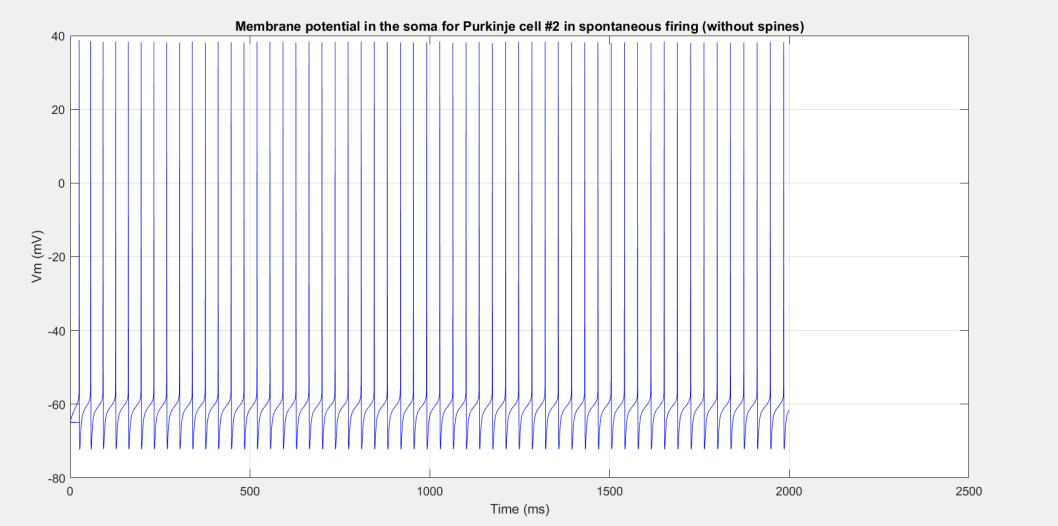


Figure 4.9: Plot of the membrane potential of the soma over time for the second model of the Purkinje cell in the spontaneous firing condition

The **Figure 4.10**, on the other hand, provides valuable insights on the activity of the second model of the Purkinje cell in the scenario in which positive current (from 0.1 to 1.5nA) is being injected in the cell.
While still enjoying a shorter initial integration period and higher values of membrane voltage during depolarization, the pattern of augmented spiking frequency over time is maintained even for this cell model, and in this case it presents three main "blocks" of different frequency, gradually increasing from a first regular block between 0 ms to 300 ms circa, to a second regular, more frequent in spikes, block between around 300 ms and 1300 ms, to a third and final regular and most frequent is spikes block between around 1300 ms until the end of the simulation.

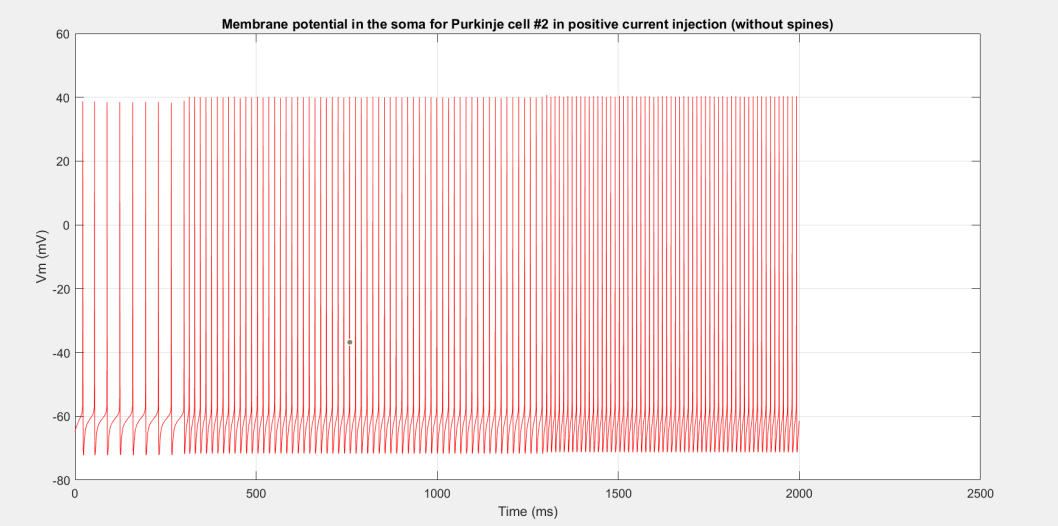


Figure 4.10: Plot of the membrane potential of the soma over time for the second model of the Purkinje cell in the positive current injection condition

These were just examples of common firing patterns to investigate, but more of them are provided in the `protocols` folder. The decision behind analyzing the spontaneous firing activity and the injection of positive current scenario were given by the availability of both simulation conditions for the two models, which allowed some degree of comparability.
By looking at the graphs of the simulations we can conclude that the main patterns remain stable across different models of the cells, which indeed makes these computational models useful and resourceful to gain insights on the behavior of these neurons. Of course we encountered some differences, but none of them were substantial to the point of not being able to recognize and compare similar patterns. Furthermore, differences are partly due to the fact that the second model investigated is to be intended as a more evolved and detailed version of the first. On the other hand, neurons in real life present differences in their spiking activity as well: it is correct to focus on patterns, but also to consider how the uniqueness of each brain is rooted in the specific activity of every cell.

While the current section was mostly concerned with the usage of the NEURON tool to investigate multi-compartmental, biologically realistic neurons, which require a good amount of computational power to simulate, we are yet to investigate local circuit dynamics by reducing each neuron to a point-like structure, and allow for more extensive simulations. Therefore, let us move on from multi-compartmental models of single cells to the reconstruction and simulation of an entire section of the cerebellar cortex, while utilizing point-like neurons and working within the BSB framework, using the NEST tool.

4.4 Cerebellar point-neuron network reconstruction and simulation within BSB and NEST

4.4.1 Aim

This section of the project work was aimed at reconstructing all the layers, namely, let us recall, the granular, Purkinje and molecular layers, of the cerebellar cortex of a mouse and at simulating its activity in a point-neuron manner, utilizing the NEST tool within the BSB framework, in order to obtain another solid element, along with what will be presented in **Section 4.5**, future research could start from, when the adequate tools will be available.

4.4.2 Development

Firstly please note that, in order to carry out this part of the project work, I was granted special access to a GitHub repository of the Brain and Behavioral Sciences laboratory of Pavia, called 'cerebellum', which is not yet available to the public. Therefore, running the following command will result in an unsuccessful attempt if you are not flagged as a member of the team via your GitHub profile beforehand. I recall that I am located in the same folder, "Anna", where all the installations took place, and my virtual ennvironment is still active.

```
git clone https://github.com/dbbs-lab/cerebellum
```

This command will clone the 'cerebellum' repository in your folder. You are going to be asked for a password, which does not correspond to the GitHub password associated to your e-mail but to a token, called the Personal Access Token (PAT), you can directly create from your GitHub profile, by navigating to 'Settings' via the profile menu, selecting 'Developer settings', and then accessing the 'Personal access tokens' section. From there, a new token can be created with the desired permissions. It is important to securely store the token upon creation, as it will not be visible again for security reasons.
When the cloning process is finished, access the newly created directory and finish the installation in editable mode, so that any changes made to the source code are immediately reflected without needing to reinstall the package. To do so, run the following commands:

```
cd cerebellum
pip install -e.
```

Reconstruction

The next step, automated by the BSB, consists in reconstructing the cerebellar network. It is enough to run the command:

```
bsb compile configurations/mouse/mouse_cerebellar_cortex.yaml -v4 --clear
```

which compiles a cerebellar cortex mouse model configuration using data from a `.yaml` file (also `.json` files are compatible), with verbose output, meaning detailed logging/output will be shown during compilation, and prior compilation data cleared to ensure a clean build. Inside the `.yaml` file one can find information about the layers to be constructed, the partitions and cell types that will be present in the network, and instructions for placement and connectivity. A reference to these commands can be found in the documentation of the BSB [1], although the examples provided refer to `.json` files.

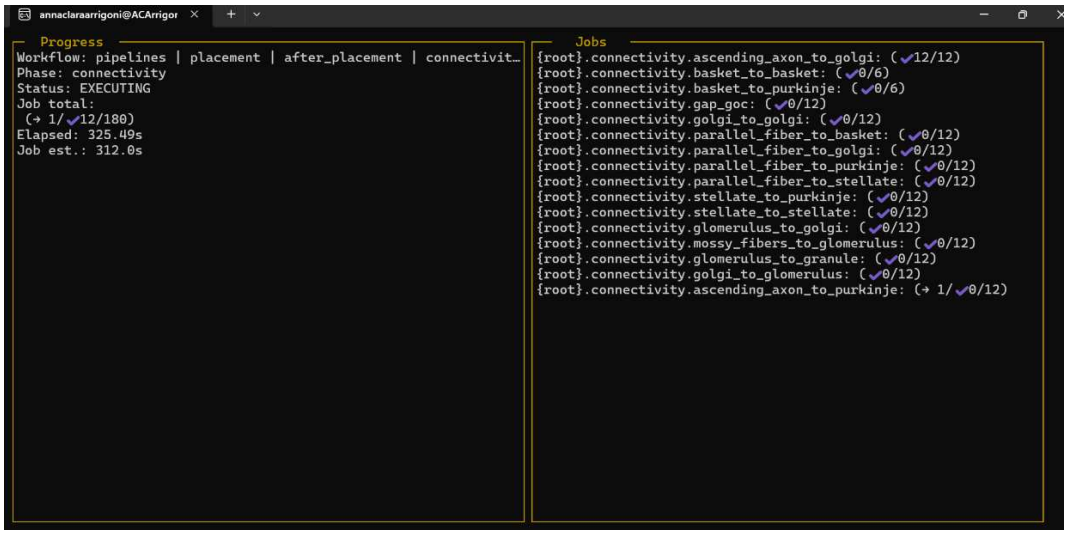


Figure 4.11: Panel shown by the BSB during the process of reconstruction

The **Figure 4.11** shows the panel that the BSB opens when the reconstruction process starts. On the top left part of it one can notice a list of all the processes handled by the BSB in this procedure. While the tool keeps track of the step in the workflow that it is addressing, it also provides an estimation of the amount of computation time for each step, while one can also monitor the specific jobs happening by looking at the right part of the screen. All the computation took around three hours on my PC.
When the process is over, it will have produced a microcircuit of the mouse cerebellar cortex and stored it in the `mouse_cerebellum.hdf5` file. This command will also automatically produce a separated `.pdf` report on the structure of the microcircuit obtained: `bsb_report_structure.pdf`.
The **Figure 4.13** and the **Figure 4.14** show a graphical representation of the reconstructed network and a colour legend one could find in the `.pdf` report. The results will be better commented and analyzed in **Section 4.4.3**.

Simulations

Firstly, some preliminary commands need to be ran, while still being in the cloned 'cerebellum' directory, in order to prepare the environment for the simulation.

```
cd cerebellum
cd nest_models
pip install appdirs
python build_models.py
```

After locating yourself in the adequate directory, install `appdirs` so that the following command, `python build_models.py`, can work successfully, loading the needed models for the BSB to work with. Now it is enough to reconfigure the previously obtained `.hdf5` file with the simulation we need, make sure to be located in the first 'cerebellum' directory. In the project I simulated in-vitro basal activity.

```
bsb reconfigure mouse_cerebellum.hdf5 configurations/mouse/nest/basal_vitro.
yaml
```

The `basal_vitro.yaml` file contains the details and parameters, expressed in NEST syntax, of the simulation we aim to produce. To check if the `.hdf5` file has been recompiled successfully, you can run: `ls -ltrh` and notice if the date and time next to the `.hdf5` file are the most recent ones, like in **Figure 4.13**.
Ultimately create a new folder to store simulation results and proceed with the simulation itself, by specifying where the output should be located, or the newly created directory, with `-o` and asking for verbosity again, if needed (`-v4`).

```
mkdir simulation_results
bsb simulate mouse_cerebellum.hdf5 basal_activity -o simulation_results -v4
```

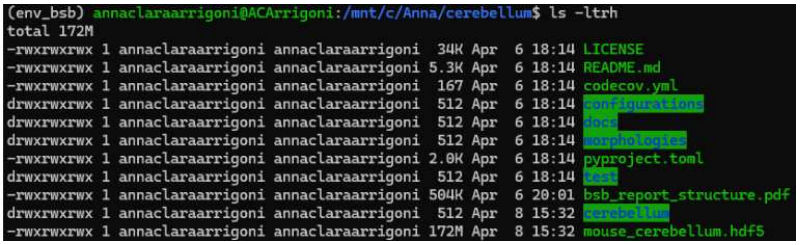


Figure 4.12: Sanity check for reconfiguration

After this step, one could visualize the simulation results by utilizing this snippet of code:

```
from neo import io
import matplotlib.pylab as plt  # you might have to run pip install matplotlib
                                # from your terminal first

# Read simulation data
my_file_name = "simulation_results/8a879296-3438-4b8e-905d-465a1b0d182a.nio"
sim = io.NixIO(my_file_name, mode="ro")
block = sim.read_all_blocks()[0]
segment = block.segments[0]
my_spiketrains = segment.spiketrains

nb_spike_trains = len(my_spiketrains)
fig, ax = plt.subplots(nb_spike_trains, sharex=True, figsize=(10, nb_spike_trains * 6))
for i, spike_t in enumerate(my_spiketrains):  # Iterate over all spike trains
    name = spike_t.annotations["device"]  # Retrieve the device name
    cell_list = spike_t.annotations["senders"]  # Retrieve the ids of the cells spiking
    spike_times = spike_t.magnitude  # Retrieve the spike times
    ax[i].scatter(spike_times, cell_list, c=f"C{i}")
```

```
    ax[i].set_xlabel(f"Time ({spike_t.times.units.dimensionality.string})")
    ax[i].set_ylabel(f"Neuron ID")
    ax[i].set_title(f"Spikes from {name}")
plt.tight_layout()
plt.savefig("simulation_results/raster_plot.png", dpi=200)
```

You can save this piece of code inside a `.py` file and launch the command:

```
python graph_results.py
```

This way, the previously stored simulation output, located in the 'simulation_results' folder, gets opened and plotted, by saving a `.png` file in the same directory, reported in the **Figure 4.16** and analyzed in **Section 4.4.3**. The image is obtained by using a Python library called 'matplotlib', which is ideal for creating graphs and plotting various kinds of data.
As it will be clearer in **Section 4.4.3** this command produces an essential graphical report of the simulated activity, but it cannot capture useful details nor does it color match the neurons in the legend of the reconstructed network (see the **Figure 4.14** for reference). In order to obtain more extensive results, supposing we are still locaded in the `cerebellum` directory, one should run the following lines, after launching the command `ipython`, which allows you to directly work within an interactive python shell:

```
from cerebellum.analysis.spiking_results import BasicSimulationReport
file = "mouse_cerebellum.hdf5"
nio_folder = "simulation_results/"
report = BasicSimulationReport(file, "basal_activity", nio_folder)
report.print_report("simulation_results/extensive_results.pdf")
```

The first line imports the `BasicSimulationReport` class, which provides methods for analyzing and summarizing simulated spiking activity, while the following two lines respectively define the `.hdf5` input file containing the simulation results, and specify the folder where the `.nio` file is located and where output will be stored. The penultimate line creates an instance of the `BasicSimulationReport` class, passing the file name, simulation protocol and final directory. While this object is used to access and analyze spiking data, the last line generates a comprehensive PDF report summarizing the simulation results and saves it to the specified location. Some of these results are reported in the **Figure 4.16**, the **Figure 4.17** and the **Figure 4.18**.

4.4.3 Results and discussion

Reconstruction

As reported in **Section 4.4.2**, the data relative to the reconstructed network are stored in a `.hdf5` file, which is hardly interpretable by the human eye since it is mostly concerned with 3D coordinates. However, as previously mentioned, in the `.pdf` report of the reconstruction, one could find a schematic representation of the cells in the cerebral cortex, as displayed in the **Figure 4.14**. We can analyze this portion of the cortex (modeling the whole cortex would be very computationally expensive) visually and notice how the BSB was able to locate all the neurons in a spatially plausible way, respecting the geometry and cell placements of each layer, considering the color-coded legend reported in **Figure 4.15**. Each dot is to intended as the soma of the neuron.
We can notice a numerous amount of (red) granule cells in the lower, or granular, layer, with (blue) Golgi cells spread out in the layer as well. Proceeding from bottom to top we see (green) Purkinje cells, distributed on an area that resembles a flat plane, almost. Ultimately, the inhibitory inteneurons: (yellow) stellate cells and (orange) basket cells are to be found in the top layer, the molecular one. Mossy fibers (in pink) are not really visible, as they are input neurons and would lay below the very thick red layer of granule cells.

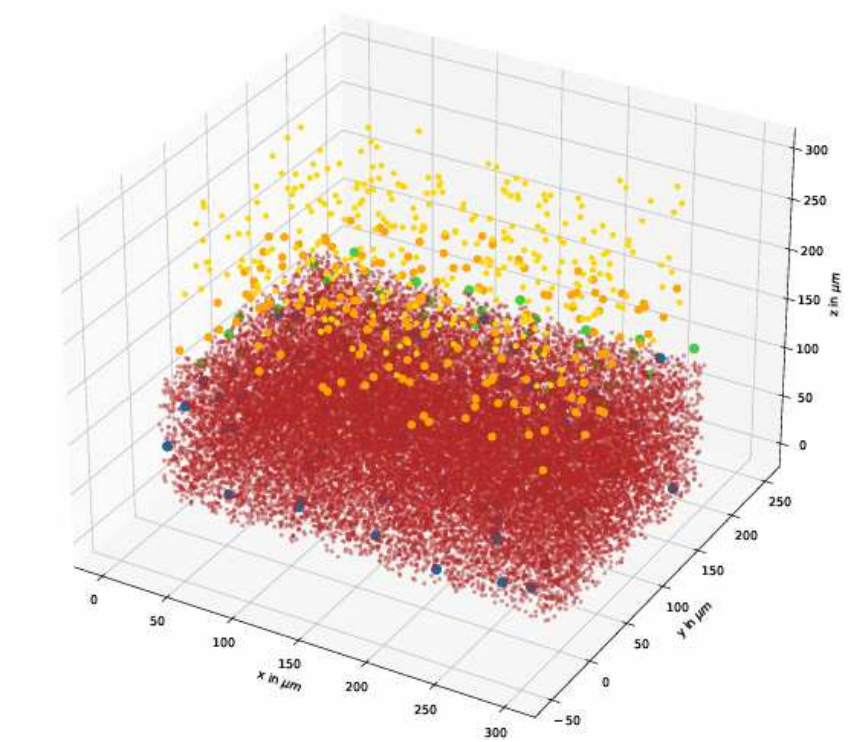


Figure 4.13: Placement of the cells of the cerebral mouse cortex

Figure 4.14: Legend of the cell types based on their colour

Simulations

As specified in the previous section, we saved the results of the simulation in a dedicated folder, namely "simulation_results", and we ran a Python script over the `.nio` file saved in there to obtain a graphical, more interpretable, representation of the firing activity of each neural population. Said picture is reported in the **Figure 4.15**. Note that, while in the picture we see a 4x2 "grid", originally, the picture was a 8x1 "grid", which I modified to fit the image in one of these pages, since it was too long to both make it fit and let it be readable. Each section of the picture detects the activity over time of specific neurons within different populations. While having very sparse background noise, granule and Purkinje cells (let us recall that Purkinje cells are authorhythmic and produce spontaneous activity to a degree) are the populations with most pronounced activity. The glomerolus also appears to be quite active. Similiar in their activity patterns are the inhibitory interneurons of the molecular layer: the stellate and basket cells, which both have an intense activity, but with more intervals compared to the previously mentioned populations. Ultimately Golgi cells and mossy fibers present a more balanced activity, alternating spikes and pauses in a more equilibrated way.
However, as previously mentioned, the **Figure 4.15** is not extremely informative nor does it provide a direct link to the cell colors assigned in the reconstruction of the network. However the extensive analysis described in precedence provided more interpretable and detailed results, contained in the **Figure 4.16**, the **Figure 4.17** and the **Figure 4.18**.

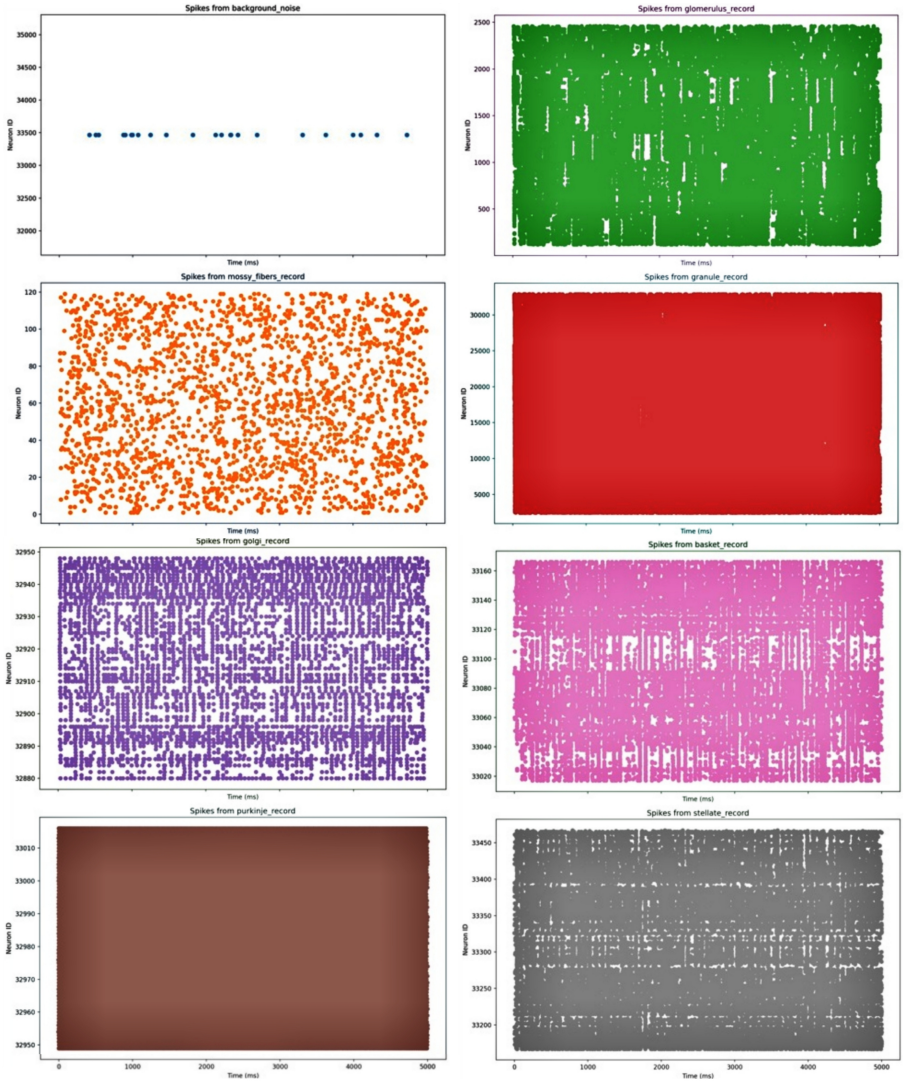


Figure 4.15: Graphical representation of the firing activity of each neural population

The **Figure 4.16** plots both the spiking activity for each neuron within a population across time and the spike counts of the latter in separated time intervals. By looking at the graphs we can once again notice how Purkinje cells are the most active ones, resulting in very intense spiking activity and high spike counts across all time bins. Granule, basket and stellate cells seem to produce a frequent and quite regular activity, followed by Golgi cells and mossy fibers, which appear to fire slightly less. The **Figure 4.17**, on the other hand reports mean firing rate across time within each population and the **Figure 4.18** reports their inter-spike-intervals (ISIs). Reasonably, more intense activity led to higher mean firing rates and vice versa, and we obtained adequate distribution of ISIs with respect to the activity of the populations (for example, shorter ISIs for intense activity).
While only these graphs were included in the present document, note that in the original PDF report also a numerical table reporting the average firing rate and ISIs of each population, an analysis in the frequency domain and a legend identical to the one of the **Figure 4.14** were provided. They were not included here for the sake of reporting only essential results.

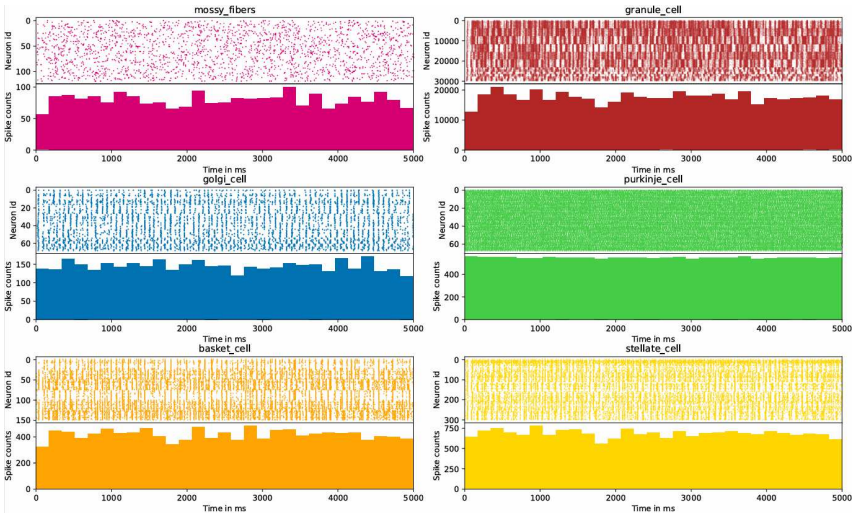


Figure 4.16: Graphical representation of the firing activity of the neural populations and their spike counts within binned time intervals

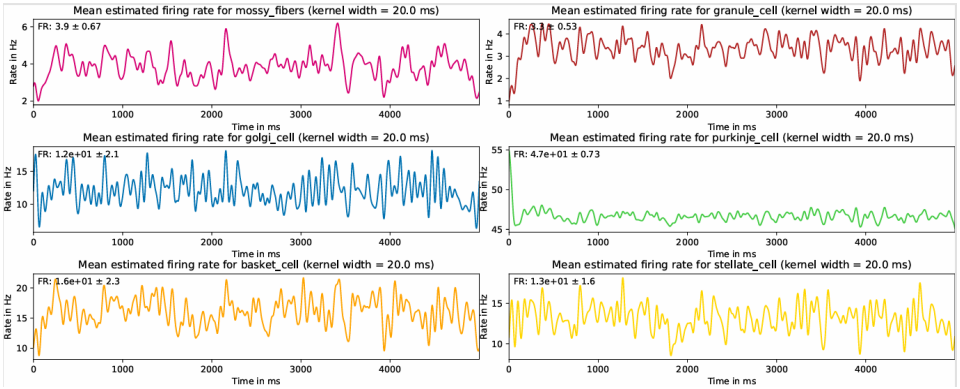


Figure 4.17: Graphical representation of the mean firing rate of the neural populations

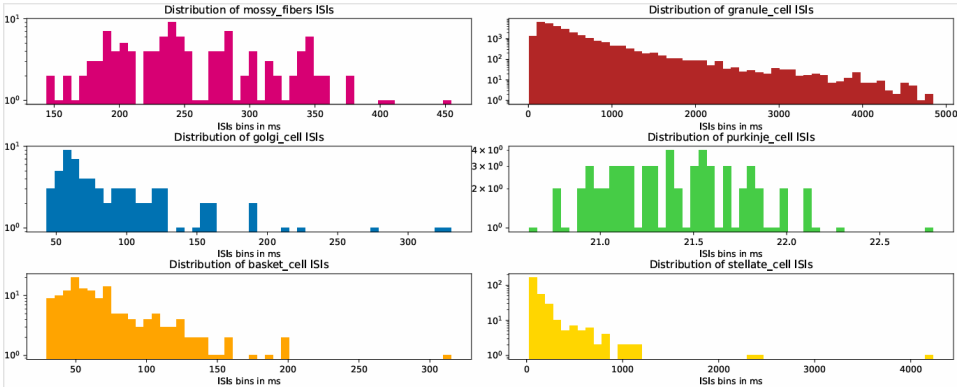


Figure 4.18: Graphical representation of the inter-spike-intervals of the neural populations

Obtaining this sort of results is relevant, as it will be better investigated in **Chapter 5**, since future research on these matters, when the right tools will be available (for example an extension of NEAT that fits the developed reduced models in a NEST circuit), will be eventually able to combine point-like neural networks, like the one developed, simulated and

investigated in this section, with not only multi-compartmental models of specific cells, but also with reduced, few-compartmental models like the one of the Purkinje neuron, final step developed for the project work, that will be presented in the following section.
As for now, purely regarding cerebellar circuits, it can be relevant to cite the repository `cerebellar-models` (in its version 0.7) which is publicly available at `https://github.com/dbbs-lab/cerebellar-models`. It is built on NEST 3.8 and NESTML 8, and addresses the strong need for an open-source, canonical cerebellar circuit model that can be easily reconfigured to include a wide range of features. The canonical olivocerebellar model was developed using the Brain Scaffold Builder (BSB). The circuit is geometrically reconstructed by specifying both cell placement and the connectome, sing the full morphologies properly oriented, when available.

4.5 Purkinje cell reduction to a few-compartmental model, construction and simulation

4.5.1 Aim

This section of the project work was aimed at reducing the morphology of the Purkinje cell, simulate its activity and compare it to the one of its respective "full" multi-compartmental model to explore the NEAT tool and its advantages. Also the granule cell was investigated using this tool, in order to have an example for a simpler morphology as well. Both cells were reduced to a somatic and a relevant dendritic compartment, chosen to be central in the dendritic trees.

4.5.2 Development

To work with NEAT, Jupyter notebooks placed in the `tutorials` directory of the GitHub cloned repository were utilized in this project. There are a couple of preliminary steps to consider before working with these notebooks. First if all you need to place yourself in the aforementioned directory, the virtual environment being active, and launch the command:

```
pip install notebook
```

which installs the classic Jupyter Notebook interface using `pip`, Python's package manager. After that, I suggest immediately running this command:

```
neatmodels install TutorialChannels -s neuron -p channelcollection_for_tutorial.py
```

which is used to install a collection of ionic channel models into the NEURON simulation environment using the `neatmodels` tool: it installs the model named `TutorialChannels`, targeting the `neuron` simulator (specified by the `-s` option), and using the definitions of the ionic channels provided in the Python file `channelcollection_for_tutorial.py` (specified by the `-p` option).
After this step it is enough to launch:

```
jupyter notebook
```

to obtain a link to operate code directly on Jupyter notebooks.

With this setup one should be able to explore the `.ipynb` notebooks already present in the repository. However in this section I will refer to two notebooks that were created during this project work in order to investigate the reduction of the granule and Purkinje cells. These notebooks can be accessed via my personal GitHub profile. This is what to search to get to the repository:

```
https://github.com/ninoonthehub/neat-jupyter-notebooks
```

As mentioned in **Section 3.3.2**, NEAT accesses morphologies through files in the standard `.swc` format. In the beginning of the development of this section of the project work, we tried to access the morphology of the Purkinje cell developed in the laboratory of Brain and Behavioral Sciences of the University of Pavia, displayed in the **Figure 4.2**. However we soon noticed that this operation was not feasible, as NEAT, at least for now, only supports `.swc` files that only include indexes of the coordinates in the range 1 to 3, where 1 indicates locations on the soma, 2 indicates locations on the axon and 3 indicates locations on the dendrites. Therefore we ended up using Miyasho's morphology of the Purkinje cell (the one shown in the **Figure 4.1**) and a modified version of the granule cell `.swc` file produced in the laboratory of Brain and Behavioral Sciences of the University of Pavia, which I included to be present in the GitHub repository of the two notebooks as well. Miyasho's Purkinje can be found in NEAT's repository in the following directory: `examples/models/morphologies`, under the name `purkinje1.swc`. In this project, i loaded both files in the `tutorials/morph` folder.

The following step in this section of the project was to carry out an analysis on the ionic channels that should have been added to the model, based on Miyasho's publication regarding his morphology of the Purkinje cell [45], the `.mod` files associated to it and the available ionic channels implemented in NEAT under the Hodgkin-Huxley paradigm. The **Table 4.1** reports the results of this analysis, featuring optimal parameters for the conductance and reversal potential of each channel.

Table 4.1: Ion channel conductances and reversal potentials

Ion Channel	Conductance	Reversal Potential (mV)
NaF	10	60
NaP	10	60
Na_Ta	$1{,}71\times10^{6}$	50
KM	0,00004	-85
KA	0,015	-85
Kh	0,0003	-30
Khh	0,036	-85
KD	0,0045	-85
KC3	0,06	-85
Kv3_1	$0{,}45\times10^{6}$	-85
K23	0,00039	-85
CaE	0,0005	135
CaT	0,0005	135
CaP2	0,0045	135
Leak	$0{,}0006\times10^{6}$	-80

In his paper, Miyasho directly mentions the ionic channels CaT, CaR, CaP, CaE, KA, KD, Kar, KM, Kdr, KC, K2, NaF, NaP and the leakage channel. NEAT did not have implemented models for CaR, Kar and Kdr, while it had models for KC3, K23, but not directly for KC and K2. Ultimately, NEAT presented some implemented channels, that were nonetheless not mentioned in Miyasho's publication, namely Na_Ta, Kv3_1, SK, Kh, Khh, h and CaP2.

In order to make the two notebooks import and use the ionic channels specified in the **Table 4.1**, which can be found in the directory of NEAT's GitHub repository `examples/models/channels` in the `channels_miyasho.py` file it was necessary to add them in two specific files, one to be found in the neat site-package of Python within the virtual environment and called `channelcollection.py`, the other in the `tutorials` directory, called

`channelcollection_for_tutorial.py`. In my case, these were the the pathways to get to them:

`mnt/c/Anna/env_bsb/lib/python3.10/site-packages/neat/channels/channelcollection/channelcollection.py`

and

`mnt/c/Anna/NEAT/tutorials/channelcollection_for_tutorial.py`

After modifying these files, it is necessary to re-run the previously cited compiling command to use the two notebooks:

```
neatmodels install TutorialChannels -s neuron -p channelcollection_for_tutorial.py
```

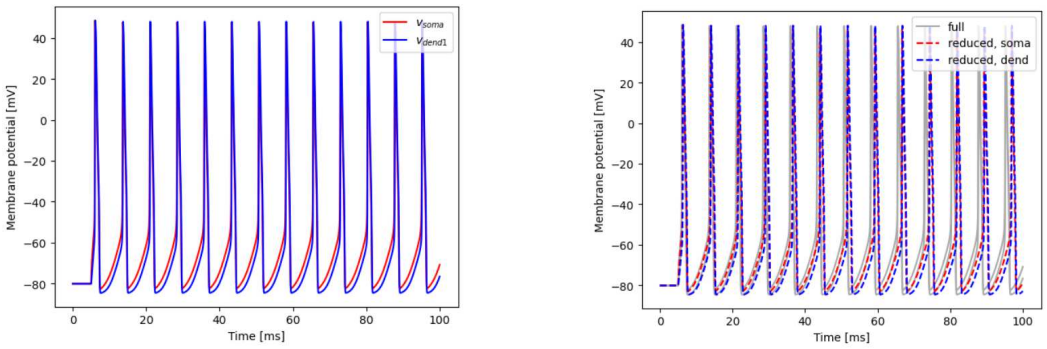
From now on one should be able to run the content of the two notebooks smoothly. Thorough explanations are provided at each code cell of the notebooks, but here a general review of the content of the two notebooks is reported, as they both follow the same structure.
First the class `PhysTree` opens the `.swc` files and stores each location of the morphology as a node object, namely a `PhysNode` one could explore the attributes of, for example the membrane capacitance or the axial resistance.
After, the adequate ionic channels, or better, the combination of ionic channels which allowed us to get a plausible activity, are imported and their parameters for conductance and reversal potential are specified following the results of the **Table 4.1**.
Some analysis and exploration of the physiological parameters is then offered both in the active and passified version of the trees. The notion of computational tree is also explored, followed by the usage of the class `NeuronSimTree` to set up a simulation for the two cells: in the case of the granule a small current injection (0.1 nA) is provided, while in the case of the Purkinje cell no current is given, in order to try to simulate authorytmic Purkinje activity, although to get some spikes it was necessary to keep at least some synaptic input, which is a limitation that will be addressed in **Section 6.2**. Some selection over locations across the tree can be done to record the activity in said places and followingly show it visually through a `matplotlib` graph.
The class `GreensTree` is then explored. It uses a mathematical tool called Koch's algorithm to analyze activity in the cells both in the frequency domain through a Fourier analysis and in the time domain in a second moment, in order to ultimately obtain some impedance matrices on the motion of the currents across the tree.
We then reach the core of the notebooks, where the class `CompartmentFitter` is used to obtain simplified compartmental models, where the parameters of the compartments are optimized to reproduce voltages at any set of locations on the morphology. One somatic compartment and one dendritic compartment are specified in each of the two cells to be the final locations to be mantained in the reduction process. Finally, the class `NeuronCompartmentTree` is utilized to simulate the activity of the reduced model and compare it to the activity of the multi-compartmental one, once again with the visual help of a `matplotlib` graph.
Ultimately, the same reduction and simulation pipeline is carried on for both cells in the case of a single-compartment reduction to see if any relevant difference arises when using few-compartmental models compared to point-neuron models.

4.5.3 Results and discussion

Granule cell

In this section a comment on the simulated activity of the granule cell is provided. More extensive exploration can be found in the `granule_cell` notebook in my GitHub repository. The **Figure 4.19 (a)** shows the simulated activity of the "full" multi-compartmental

granule cell recorded at the soma and at one dendritic compartment. Recall that this activity is generated by providing a current injection in the soma with amplitude of 0.1 nA, a delay of 5 ms and a duration of 100 ms. A graph that differentiates the activity of the soma and the dendrite can be found in the notebook. We can see how in both compartments the activity is steady and homogeneous across the simulation, although a bit too frequent with respect to empirical biological recordings of granule cells.
The **Figure 4.19 (b)**, on the other hand, compares the activity of the same "full" multi-compartmental granule cell model to its reduced version under the same simulation paradigm. By looking at it we can appreciate how the reduced model, faster in computation due to its simplified structure and smaller size, still captures the original activity quite faithfully.



(a) Spiking activity of the granule cell

(b) Comparison between the spiking activity of the multi-compartmental and few-compartmental granule cell

Figure 4.19: Activity for the granule cell

The **Figure 4.20**, ultimately, compares the activity of the "full" multi-compartmental granule cell model to a point-neuron version of it, where only the somatic compartment was included, under the same simulation paradigm. By looking at it we can notice that, in this case, the activity is still captured fairly by the point-neuron model, meaning that the few-compartmental model does not make much of a difference in this specific case with a very small cell.

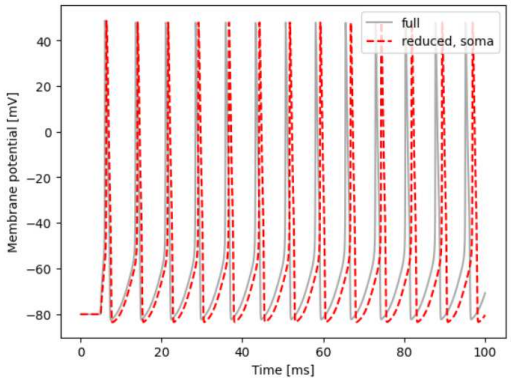
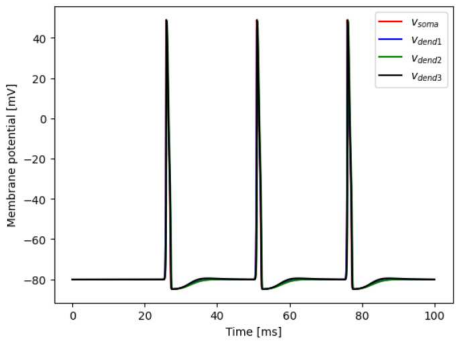


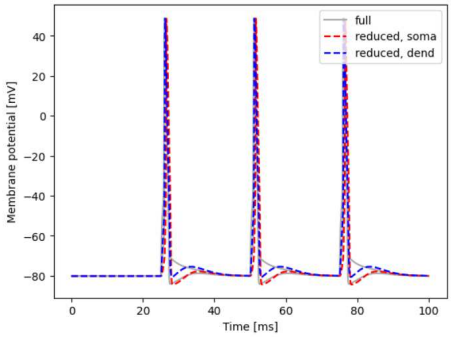
Figure 4.20: Comparison between the spiking activity of the multi-compartmental and point-neuron granule cell

Purkinje cell

In this section a comment on the simulated activity of the Purkinje cell is provided. More extensive exploration can be found in the `purkinje_cell` notebook in my GitHub repository. The **Figure 4.21 (a)** shows the simulated activity of the "full" multi-compartmental Purkinje cell recorded at the soma and at three dendritic compartments. This activity in the notebook is generated by providing a synaptic input in a random dendrite with rise and decay times of 0.2 ms and 3 ms respectively, and a reversal potential of 0 mV. A graph that differentiates the activity of the soma and the three dendrites can be found in the notebook. We can still see how in all the compartments the activity is homogeneous across the simulation, although we could not get the Purkinje cell to produce activity in a fully autorhythmic way, because at least some synaptic input had to be specified in order for it to spike.
The **Figure 4.21 (b)**, on the other hand, compares the activity of the same "full" multi-compartmental Purkinje cell model to its reduced version under the same simulation paradigm. By looking at it we can once again appreciate how the reduced model, faster in computation due to its simplified structure and smaller size, still captures the original activity quite faithfully.



(a) Spiking activity of the Purkinje cell



(b) Comparison between the spiking activity of the multi-compartmental and few-compartmental Purkinje cell

Figure 4.21: Activity for the Purkinje cell

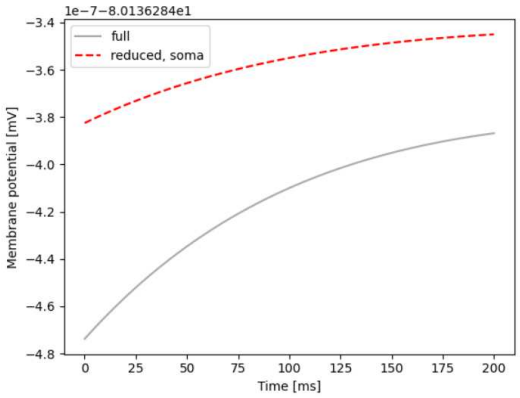


Figure 4.22: Comparison between the spiking activity of the multi-compartmental and point-neuron Purkinje cell

To conclude, the **Figure 4.22**, compares the activity of the "full" multi-compartmental Purkinje cell model to a point-neuron version of it, where only the somatic compartment was

included, under the same simulation paradigm, but removing any support spike train. By looking at it we can notice that, in this case, the original activity is not being reproduced faithfully. We can understand that, when using NEAT for models of cells of bigger sizes with respect to the granule cell, like the Purkinje cell, reduced few-compartmental models supported by a spike train generation leads both to more plausible activity and better reproduction of the original activity than in the point-neuron case. The fact that without a spike train the Purkinje cell is not able to fire in a biologically plausible way remains a limitation, though.

4.6 Wrapping up

In this section of the thesis, after a description on how to set up the right environment for the project, we have explored how to visulize the morphologies of Purkinje cells and how to simulate their activity using the NEURON simulator to gain insights on multi-compartmental models. Then, a section on the use of the BSB interfaced with the NEST simulator was provided, in order to show how to reconstruct and simulate different pupulations of neural cells in a point-neuron manner within a portion of the cerebellar cortex. Finally, a work on reduced, few-compartmental models using the NEAT tool was reported, and trough two Jupyter notebooks providing examples for the granule and Purkinje cell, the advantages of adopting these third kind of neuron models, borrowing biological realism from multi-compartmental models and computational efficiency from point-neuron models, became evident, above all in the case of the bigger Purkinje cell. While future research will find ways to merge these approaches, for example by inserting few-compartment models inside point-neuron networks, this chapter was aimed at providing an hopefully exhaustive and experimental review of these three different modeling approaches, especially in the usecase of the Purkinje cell.

Chapter 5
Medical and ethical context of the present work, and its future perspectives

5.1 Introduction

While the first chapters, from addressing biological evidence, needed to navigate the features of the work, to providing methodological insights to capture the modeling framework adopted, were aimed at presenting the technical context of the project reported in **Chapter 4**, the current chapter offers a more ample overview of the medical, ethical and social contexts of the present work, aiming to provide a complete vision which integrates the previously presented content with broader implications.
While keeping a very close look to the medical context and the ambit of research, the aim behind this section of the present document is also to hint to the social relevance of the field of study this project is set within, and to highlight some ethical issues that need to be considered while talking about it.

5.2 Personalized medicine

A very reasonable field of application of the present work is personalized medicine, or precision medicine.
Personalized medicine refers to an approach in healthcare aimed at tailoring medical treatment of various natures to the individual characteristics of each patient, such as their genetic profile, physiological traits, lifestyle, and environment [18].
As it is now possible to digitize a human being in its entirety, by relying on the integration of multiscale data, including demographics, biosensors to capture the individual's physiome, imaging to depict their anatomy, etc., it is clear how neural models easily come into play, since in this setting they can contribute to the understanding of the individual differences of the nervous systems of patients, for example by understanding differences in neural responses to specific drugs or alterations in neuropathological cases by using patient-specific simulations of their reconstructed networks. Many benefits come from adopting this approach, such as obtaining more accurate diagnoses, tailor-made therapeutical plans and side effects reduction. However, this promising approach also raises some ethical concerns, such as equitable access to this kind of services and following treatments, as their very nature could make them exclusive to a restricted amount of people, and data privacy, concerning how personal and genetic data are handled by the institutes that practice these services [22].

In the next section, some recent computational neuroscience projects which can fall within the broader ambit of precision medicine will be addressed.

5.3 Research projects

In recent years, several large-scale neuroscience initiatives have been launched with the aim of deepening our understanding of the human brain through integrative and computational approaches. Among these, the *Human Brain Project*, funded by the European Commission in 2013, sought to simulate brain function by combining neuroscience data with high-performance computing through large-scale data integration and modeling. It combined neuroscience, medicine, and computing to create digital tools and infrastructures, such as EBRAINS, to support brain research across Europe [27]. Closely linked to it, the *Blue Brain Project*, based at EPFL in Switzerland, and started in 2005, focused on reconstructing and simulating neocortical microcircuits at a cellular level [8]. Across the Atlantic, the *BRAIN Initiative*, launched in the United States in 2013 by the Obama administration, has promoted the development of novel neurotechnologies to map and manipulate brain activity, specifically focusing on bridging the gap between observable behavior and neural mechanisms [9].
More recently, projects like *EBRAINS 2.0* (2024-2026) have emerged as successors to the *Human Brain project*, aiming to consolidate a pan-European infrastructure for neuroscience research based on openness, collaboration and reproducibility [16]. In parallel, the *Virtual Brain Twin* initiative (2024-2027) proposes the creation of individual, biologically-informed digital replicas of human brains, which incorporate a person's specific anatomy and function. These computational "twins" are expected to play a decisive role in personalized medicine, enabling predictive simulations for diagnostic and therapeutic purposes, disease progression monitoring and treatment outcomes [13].
The **Figure 5.1** shows the logos of all the mentioned initiatives.

Figure 5.1: Logos of the research projects mentioned

Beyond the differences, some common principles are shared among these projects: interdisciplinarity and massive data collection to tackle knowledge fragmentation on the brain, multiscale investigation of neural processes both in space and time, applications to neurological diseases, exploitation of ICT resources for data processing and development of new brain-inspired technologies.

Ultimately, I include a mention to a future project, that will based between Italy and the United Kingdom, which one of my supervisors had the occasion to assist to the presentation of, as it will be a collaboration between the University of Pavia and the University

College London (UCL). Within this project, the name of which is not yet disclosed, the NEAT tool [50] will be used to investigate the role of sodium in multiple sclerosis through computational neuron models. As presented in **Section 3.3.2**, by using NEAT, we are able to create simplified neuron models with few compartments that allow the investigation of specific ionic channels involved in the pathology. Starting from a Purkinje neuron model, the researchers will extend the study to other cerebellar neuron types and to the cerebellar microcircuits. This project highlights the numerous applications of modeling in pathological conditions and, as a matter of fact, a more in-depth analysis on the condition of autism will be given as an extensive example of a pathological condition in the next section.

5.4 Pathological conditions: the autism IB2 KO case in mice

As it has been mentioned in the two previous sections, one of the most prominent fields of application of digitalized models of brain structures is the attempt to reconstruct and simulate neuropathological conditions. In the current section a specific example will be addressed: the possibility to model the cerebellar circuit in mice brains for mice with the IB2 KO model of autism.

5.4.1 The Austistic Spectrum Disorder

Let us start by providing a very general overview of this condition, by highlighting the main traits exhibited by autistic individuals, supported by medical and scientific evidence.

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition characterized by persistent difficulties in social communication and interaction, alongside restricted, repetitive patterns of behavior, interests, or activities. The term "autism" was first introduced in the early 20th century by the swiss psychiatrist Eugen Bleuler [7], though his reasearch pertained more to the ambit of schizophrenia and it was not until the 1940s that Leo Kanner and Hans Asperger (note that another common way to refer to the ASD is by using the expression "Asperger's syndrome") described children with atypical social development and communication, laying the foundation for contemporary understanding of the spectrum [29, 4]. Today, ASD is recognized as a broad continuum of conditions that vary widely in severity and presentation. Core traits may include challenges in interpreting social cues, delayed language development, and an intense focus on specific interests [36]. Medical and neuroscientific research has revealed a strong genetic component, with numerous associated genes involved in synaptic development and neural connectivity [5], and offered some unresolved speculations on overpruning or underpruning of neural connections in autistic individuals during brain developement [65]. Neuroimaging studies have also identified atypical patterns in brain structure and function, particularly in regions linked to social cognition, such as the amygdala and prefrontal cortex [5]. While no single cause has been identified, ASD is understood as resulting from a complex interplay of genetic, epigenetic, and environmental factors. Current diagnostic criteria, as defined in the DSM-5, emphasize the dimensional nature of the disorder, acknowledging the diversity of experiences and abilities among autistic individuals [36].

Autism in the cerebellum

Various evidence supports the fact that the role of the cerebellum in defining the mechanisms underlying the autism spectrum disorder is a non-marginal one. For example, various neuroimaging studies and neurobiological research has has shown abnormalities in neurotransmitter systems and cerebellar circuitry in individuals with autism, while anatomical studies detected differences in volume, morphology and connectivity of the cerebellum between individuals with ASD and neurotypical ones [6]. For example, studies on MRI data have been conducted, highlingting significant stuctural abnormalities in

the developement of the cerebellar cortex, which resulted to be less thick in autistic subjects, especially in some lobes [37]. Regarding typical ASD symptoms, such as sensory over-responsivity, another study was carried on. By investigating the functional connectivity of the sensory-motor cerebellum in autism, it was shown that not only the connectivity within the cerebellum was altered, but also the connective paths to other brain regions. As a matter of fact the study showed increased connectivity with the hippocampus, the thalamus and the cortical sensory-motor regions of the cerebrum. Furthermore, the study reported reduced connectivity between the cerebellum and cerebral areas involved in socio-emotional functions, once again aligning with the hypothesis of the involvement of the cerebellum in determining ASD symptoms, such as over-responsivity or reduced socio-emotional skills [10]. Ultimately, other evidence was provided on the side of the investigation of the underlying structures impacting on the social abilities of autistic individuals: a study on MRI data found a significant correlation between their social communication abilities, assessed with an adequate measure called "the Social Responsiveness Scale (SRS)" and their cerebellar structure, using linear mixed models and canonical correlation analysis, by performing cerebellar parcellation and looking at the activity of the cerebellum in specific areas [17].
It should now be evident how central the cerebellum is with respect to its involvement in the neural mechanisms that underpin the ASD condition and, as a matter of fact, at the moment the laboratory of Brain and Behavioral Sciences of the University of Pavia is working on developing a model for the autistic cerebellar network within the BSB framework. In the next section, we are to focus on the IB2 KO variant of autism in mice specifically, to gain further insights on the matter.

The IB2 gene

A gene is a unit of hereditary information that occupies a fixed position (its "locus") on a chromosome and that achieves its biological effects by directing the synthesis of proteins. Specifically, the IB2 gene, predominantly expressed in the brain and pancreatic tissues, and "officially" known as MAPK8IP2 (Mitogen-Activated Protein Kinase 8 Interacting Protein 2) and also referred to as Islet-Brain 2 (IB2) or JIP2, encodes a scaffold protein involved in the regulation of the c-Jun N-terminal kinase (JNK) signaling pathway, which plays a crucial role in neuronal development and synaptic signaling, and facilitates efficient signal transduction [47].
Clinical research has shown that deletions in the terminal region of chromosome 22q, particularly involving the SHANK3 gene, are associated with Phelan-McDermid syndrome (PMS) and autism spectrum disorders (ASD), and that in almost all such cases, these deletions also encompass the closely linked IB2 gene. Given that IB2 is broadly expressed in the brain and is especially enriched at postsynaptic densities, its deletion has significant neurodevelopmental consequences. In fact, experimental disruption of the IB2 gene in mice models has been shown to impair glutamatergic neurotransmission by reducing AMPA receptor-mediated currents and enhancing NMDA receptor-mediated signaling in the cerebellum (both AMPA and NMDA are excitatory glutamateric ionotropic receptors). These changes are accompanied by alterations in Purkinje cell dendritic morphology and result in motor and cognitive deficits reminiscent of autism-like phenotypes [21]. The involvement of IB2 in synaptic function and neuronal signaling highlights its potential contribution to the neurological symptoms observed in PMS and ASD, including intellectual disability, delayed language development, and autistic behaviors. For autism in particular, it has been shown that transgenic mice presenting such IB2 gene alteration, resulted in IB2 KO granule cells showing a larger NMDA receptor-mediated current and enhanced intrinsic excitability, raising the excitatory/inhibitory balance. Furthermore, the spatial organization of granular layer responses to mossy fibers shifted from a "Mexican hat" to a "stovepipe hat" profile, with stronger excitation in the core and weaker inhibition in the surround. Finally, the size and extension of long-term synaptic plasticity were remarkably increased. These results show that hyperexcitability and hyperplasticity disrupt

signal transfer in the granular layer of IB2 KO mice, supporting cerebellar involvement in the pathogenesis of ASD. [61]

5.4.2 Implications for modeling

It should be now evident how prominent the role of the cerebellar network is also when dealing with autistic brains. As a consequence it is reasonable to argue that computational models of the brain of autistic individuals should include this region and be tuned appropriately in order to address the alterations in current transmission, excitability and plasticity higlighted in the previous paragraph. For example, developers should address the changes in currents mediated by AMPA and NMDA receptors by reducing the former and enhancing the latter, tune granule cells in accordance with evidence on their hyperexcitability, adapt the response of mossy fibers to fit the less inhibited activity of the columnar sorroundings and test for coherence with the results on plasticity over time when simulating the network.
The perspectives and possibilities of modeling the autistic cerebellum are numerous, but to mention just a few, the medical community could get more insights on the underlying biophysiological mechanisms of ASD via a top-down approach which features computational models of this brain region, simulations to test medications without massive risks could be carried out, and the research community could get a step further towards personalized medicine and adopt subject-specific models of neural circuits. Of course, these scenarios will not have immediate application, and we recall that models for now are mostly based on data obtained through experiments carried out on rodents (for the most part) and other animals. While the next section will address the limits and ethical concerns around experiments involving animals, such as their translatability to the human neural domain, we conclude the present one by stressing how, although not instant, these developments could be significative and, once again, by highlighting how versatile and central the cerebellum appears to be for modeling in both neurotypical and neurodivergent settings.

5.4.3 The three R's

As mentioned in the previous section, autism in the IB2 KO case was studied on mice models first. In the study of complex neurodevelopmental conditions such as autism, preliminary research is often conducted using animal models, particularly rodents, before extending investigations to human subjects. This practice is largely dictated by ethical and practical considerations, as it allows for invasive and controlled experimentation that would not be permissible in humans. Nonetheless, the use of animals in neuroscience research remains a topic of active ethical debate. While such models provide valuable insights into brain function and pathology, questions persist regarding animal welfare, the translatability of results to humans, and the moral justification for their use. This is a further reason why modeling and simulating the brain is advantageous: not only these practices offer the possibility to study diseases in in-silico experiments and improve the validation of data and experiments with computational methods, but they also reduce the need for animal experiments, as in the long term these in-silico approaches could eventually contribute to aim of the so called 3R's of animal experiments: their reduction in terms of necessity and frequency, their refinement in terms of techniques used and invasiveness and their, ideal, final replacement with alternative methods [48].

Chapter 6
Conclusions

This is the closing chapter of my thesis. In this section, after briefly reviewing the content of the present document and the obtained results, the main limitations and future work that this project lays the foundations for are highlighted.

6.1 Review of Project Goals

This thesis was firstly aimed at understanding the relevance of the cerebellum as a region of the nervous system to be modeled in order to have a complete view of the neural framework of an individual. By showing how highly connected this region is, by presenting its numerous functions, ranging from posture and balance maintenance, to movement planning and control, to its involvement in high level cognitive functions, to social and affective behaviors, and also by providing a review of the attempts to model this brain region, in **Chapter 2** we concluded that when investigating or modeling one's neural framework this region should not be neglected.
Chapter 3, on the other hand, was aimed at presenting all the tools that were then utilized during the project work carried on during this academic year and documented in **Chapter 4**, and their main characteristics. The NEURON simulator was presented as a tool for the visualization of multi-compartmental morphologies and their simulation, followed by the BSB, a versatile tool for both the reconstruction of neural networks and their simulation, integrating simulators at different scales. Ultimately NEST, a simulator for point-neuron models was mentioned, with a specific focus on a tool part of its ecosystem, namely NEAT, especially useful for carrying out automatic simplifications of multi-compartmental structures. Finally, other automatic simplification tools were presented, let us cite the Rösssert-Pozzorini pipeline and AdEx models.
As for the core of my thesis, or the project work, it was divided in sections with respect to the main tool that was utilized in each of them. In the first section we focused on the creation of a Python virtual environment, running on Python 3.10 and accessible from the Linux Subsystem for Windows, that contained all the tools needed for the project. The following part was dedicated to the NEURON simulator. Here we investigated the differences between the morphologies of Miyasho's multi-compartmental model of the Purkinje cell and one coming from the laboratory of Brain and Behavioral Sciences of the University of Pavia, enjoying how naturally different neurons within the same population can be. Followingly, we simulated the activity of two different models of the Purkinje cell produced in the laboratory of Brain and Behavioral Sciences of the University of Pavia under two main conditions: spontaneous activity and positive current injections, comparing similiar patterns between the two in each of the scenarios.
In the penultimate part we gained insights on how a point-neuron network, especially for the cerebellar cortex, can be constructed and simulated using the BSB interfaced with the

NEST simulator, and analyzed the activity of each of the populations that were placed by the BSB in the network.
Finally we dedicated the last part to the exploration of the NEAT tool, especially aimed at investigating the advantages of utilizing few-compartmental models for specific cells. Two Jupyter notebooks have been presented: one for the granule cell and the other for the Purkinje cell. We noticed how faithful reduced models were in simulation when compared to "full" multi-compartmental ones and how in bigger cells these reduced models perform better than the classical point-neuron ones, although some limitations remain.
Ultimately, **Chapter 5** was aimed at presenting the context of the present work, ranging from medical applications in personalized medicine, to research projects across the world, to pathological use cases such as autism in the IB2 KO condition in mice and multiple sclerosis, to its possible contribution to the reduction of animal experiments.

6.2 Limitations

As previously mentioned, some limitations in this work remain.
First of all more simulation protocols could have been tested with the NEURON simulator, but this work mentioned just two for the sake of essentiality and comparability.
More of the BSB framework could have been investigated as well, but once again, this work was bounded by the thesis time and length constraints.
Overall, the NEAT fragment of the project work resulted in being the most problematic one, though. First of all not all the "ideal" channels reported in the **Table 4.1** were used, but only those functional to get a plausible activity. Furthermore the granule cell presented an elevated firing sequence, which can however be handled by setting some specific spike trains (for now, at least). The Purkinje cell, on the other hand, needed to keep at least a synaptic input or a spike train to fire even in the spontaneous activity condition, which contradicts its autorhythmic nature. For all of these limitations, however, more extensive work should reasonably lead to improved results.

6.3 Future Work

This thesis lays the foundations for future work in some possible directions. First of all a very reasonable expectation would be to export NEAT's reduced models into point-neuron NEST circuits, and to simulate them. The most recent update towards this goal has been disclosed by Willem Wybo in the documentation of the NEAT tool [69]. The researcher developed a class called `NestCompartmentTree` which allows to export a reduced model obtained within NEAT into NEST, and then handle and simulate it using NEST's syntax, but still at single cell level. Insertion in a point-neuron circuit is expected to be the next step, followed by the creation and simulation of circuits of reduced models.
Then, of course, pathological cases could be investigated. As previously mentioned, a model for the autistic cerebellar network within the BSB framework is currently being developed in the laboratory of Brain and Behavioral Sciences of the University of Pavia, while in a future project whose name is not yet disclosed, which will be a collaboration between the University of Pavia and the University College London (UCL), the NEAT tool will be used to investigate the role of sodium in multiple sclerosis through computational neuron models. Finally, while I stress once again how the adoption of few-compartmental models could contribute in moving forward towards the three R's of animal experiments, all the other scenarios mentioned in **Chapter 5** are also reasonable application fields: medicine, especially precision medicine and worldwide research projects could make use of reduction tools such as NEAT and integrate them with other state-of-the-art tools in numerous ways.

This section concludes the whole thesis. By starting from the investigation of the cerebellum through its structure, connections, numerous functions and role in modeling, moving to the presentation of a list of different modeling tools, to then show an empirical use of them in my project work and some possible applications of it in different ambits, my hope is to have overall provided a many-sided analysis on the usage and application of different modeling approaches for the Purkinje cell, a very relevant neuron of the cortex of the cerebellum, a powerful region of the brain.

References

[1] BSB documentation, BSB official site by the department of brain and behavioral sciences of the University of Pavia, 2023. (Cited on pages 18 and 39)

[2] N. Ahmadian, K. van Baarsen, M. van Zandvoort, and P. A. Robe. The cerebellar cognitive affective syndrome: a meta-analysis. *PubMed*, 18:941–950, 2019. (Cited on page 13)

[3] G. P. D. Argyropoulos, K. van Dun, M. Adamaszek, M. Leggio, M. Manto, M. Masciullo, M. Molinari, C. J. Stoodley, F. Van Overwalle, R. B. Ivry, and J. D. Schmahmann. The cerebellar cognitive affective/schmahmann syndrome: a task force paper. *PubMed*, 19(1):102–125, February 2020. (Cited on page 13)

[4] Hans Asperger. Autistic psychopathy in childhood. In Uta Frith, editor, *Autism and Asperger Syndrome*, pages 37–92. Cambridge University Press, 1991. Originally published in 1944 as *Die Autistischen Psychopathen im Kindesalter*. (Cited on page 53)

[5] R. Bhandari, J.K. Paliwal, and A. Kuhad. Autistic spectrum disorders: A review of clinical features, theories and diagnosis. *Neuroscience Research*, 2015. (Cited on page 53)

[6] M. S. Biswas, S. K. Roy, R. Hasan, and M. M. U. Pk. The crucial role of the cerebellum in autism spectrum disorder: Neuroimaging, neurobiological, and anatomical insights. *Health Science Reports*, 7(7):e2233, 2024. (Cited on page 53)

[7] Eugen Bleuler. *Dementia Praecox or the Group of Schizophrenias*. International Universities Press, 1911. Originally published in German as *Dementia Praecox oder Gruppe der Schizophrenien*. (Cited on page 53)

[8] Blue Brain Project. Blue brain project - epfl, 2025. (Cited on page 52)

[9] BRAIN Initiative. The brain initiative: Brain research through advancing innovative neurotechnologies, 2025. (Cited on page 52)

[10] M. E. Cakar, N. J. Okada, K. K. Cummings, J. Jung, S. Y. Bookheimer, M. Dapretto, and S. A. Green. Functional connectivity of the sensorimotor cerebellum in autism: associations with sensory over-responsivity. *Frontiers in Psychiatry*, 15:1337921, 2024. (Cited on page 54)

[11] Daniele Caligiore, Giovanni Pezzulo, Gianluca Baldassarre, Andreea Cristina Bostan, Peter L. Strick, Kenji Doya, Rick C. Helmich, Michiel Dirkx, James Houk, Henrik Jorntell, Angel Lago-Rodriguez, Joseph M. Galea, R. Chris Miall, Traian Popa, Asha Kishore, Paul F. M. J. Verschure, Riccardo Zucca, and Ivan Herreros. Consensus paper: Towards a systems-level view of cerebellar function: the interplay between cerebellum, basal ganglia, and cortex. *PubMed*, 16(1):203–229, February 2017. (Cited on page 12)

[12] S. Casali, M. Tognolina, D. Gandolfi, J. Mapelli, and E. D'Angelo. Cellular-resolution mapping uncovers spatial adaptive filtering at the rat cerebellum input stage. *Communications Biology*, 3(1):635, October 2020. (Cited on page 14)

[13] Virtual Brain Twin Consortium. Virtual brain twin for personalized treatment of psychiatric disorders, 2025. (Cited on page 52)

[14] DBBS Lab. Dbbs lab github repository. `https://github.com/dbbs-lab`, 2025. (Cited on page 30)

[15] R. De Schepper, A. Geminiani, S. Masoli, et al. Model simulations unveil the structure-function-dynamics relationship of the cerebellar cortical microcircuit. *Communications Biology*, 5:1240, 2022. (Cited on pages 14 and 18)

[16] EBRAINS Consortium. Ebrains 2.0 project, 2025. (Cited on page 52)

[17] Y. Elandaloussi, D. L. Floris, P. Coupe, et al. Understanding the relationship between cerebellar structure and social abilities. *Molecular Autism*, 14:18, 2023. (Cited on page 54)

[18] European Commission. Personalised medicine, 2025. (Cited on page 51)

[19] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. (Cited on page 21)

[20] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007. (Cited on page 21)

[21] Jennifer Giza, Matthew J. Urbanski, Francesca Prestori, Bipan Bandyopadhyay, Amelia Yam, Victor Friedrich, Katherine Kelley, Egidio D'Angelo, and Mitchell Goldfarb. Behavioral and cerebellar transmission deficits in mice lacking the autism-linked gene islet brain-2. *Journal of Neuroscience*, 30(44):14805–14816, 2010. (Cited on page 54)

[22] Lori H. Goetz and Nicholas J. Schork. Personalized medicine: motivation, challenges, and progress. *PubMed*, 109(6):952–963, 2018. (Cited on page 51)

[23] Duane E. Haines and M. D. Ard. *Fundamental Neuroscience For Basic And Clinical Applications*. Elsevier: Saunders, Philadelphia, PA, 4 edition, 2013. (Cited on page 19)

[24] Linxuan He, Yunhui Xu, Weihua He, Yihan Lin, et al. Network model with internal complexity bridges artificial intelligence and neuroscience. *Nature Computational Science*, 4:584–599, August 2024. (Cited on page 5)

[25] Michael L. Hines and Nicholas T. Carnevale. Neuron simulation environment, 2025. (Cited on page 18)

[26] 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):500–544, 1952. (Cited on page 22)

[27] Human Brain Project. Human brain project, 2025. (Cited on page 52)

[28] Leto K, Arancillo M, Becker EB, Buffo A, Chiang C, Ding B, Dobyns WB, Dusart I, Haldipur P, Hatten ME, Hoshino M, Joyner AL, Kano M, Kilpatrick DL, Koibuchi N, Marino S, Martinez S, Millen KJ, Millner TO, Miyata T, Parmigiani E, Schilling K, Sekerkova G, Sillitoe RV, Sotelo C, Uesaka N, Wefers A, Wingate RJ, and Hawkes R. Consensus paper: Cerebellar development. *PubMed*, 2016. (Cited on page 8)

[29] Leo Kanner. Autistic disturbances of affective contact. *Nervous Child*, 2(3):217–250, 1943. (Cited on page 53)

[30] Christof Koch. *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press, 1999. (Cited on page 13)

[31] Leonard F. Koziol, Deborah Budding, Nancy Andreasen, Silvia D'Arrigo, Sara Bulgheroni, Hiroshi Imamizu, Masao Ito, Mario Manto, Cherie Marvel, Kimberley Parker, Giovanni Pezzulo, Narender Ramnani, Domenico Riva, Jeremy Schmahmann, Larry Vandervert, and Tadashi Yamazaki. Consensus paper: the cerebellum's role in movement and cognition. *PubMed*, 13(1):151–177, February 2014. (Cited on page 12)

[32] Yoshihisa Kurachi and R. Alan North. Ion channels: their structure, function and control â an overview. *PubMed*, 554(Pt 2):245–247, January 2004. (Cited on page 20)

[33] James R. Lackner. Vestibular system: A revolution in understanding a neglected sensory system. *Current Biology*, 35(4):R150–R152, 2025. (Cited on page 11)

[34] S.Y. Lara-Aparicio, A.J. Laureani-Fierro, C. Morgado-Valle, L. Beltran-Parrazal, F. Rojas-Duran, L.I. Garcia, R. Toledo-Cardenas, M.E. Hernandez, J. Manzo, and C.A. Perez. Latest research on the anatomy and physiology of the cerebellum. *Neurology Perspectives*, 2(1):34–46, 2022. (Cited on pages 9 and 10)

[35] Roberto Lent, Frederico A C Azevedo, Carlos H Andrade-Moraes, and Ana V O Pinto. How many neurons do you have? some dogmas of quantitative neuroscience under revision. *European Journal of Neuroscience*, 35(1):1—9, 2012. Epub 2011 Dec 13. (Cited on page 7)

[36] Catherine Lord, Mayada Elsabbagh, Gillian Baird, and Jeremy Veenstra-Vanderweele. Autism spectrum disorder. *The Lancet*, 392(10146):508–520, 2018. (Cited on page 53)

[37] Qian Lu, Jun Chen, Yujie Wang, Liang Huang, Ziqiang Jiang, Bwalya A. Nguchu, Shuguang Chen, Baojuan Qiu, and Xiaohong Wang. Cerebellar structural abnormality in autism spectrum disorder: A magnetic resonance imaging study. *Psychiatry Investigation*, 20(4):334–340, 2023. (Cited on page 54)

[38] Mario Manto, James M. Bower, Adriana B. Conforto, Jose Maria Delgado-Garcia, Sonia N. da Guarda, Marcus Gerwig, Christophe Habas, Nobuhiro Hagura, Richard B. Ivry, Peter Marien, Marco Molinari, Eiichi Naito, Dennis A. Nowak, Nordeyn Oulad Ben Taib, Denis Pelisson, Claudia D. Tesche, Caroline Tilikete, and Dagmar Timmann. Consensus paper: roles of the cerebellum in motor control–the diversity of ideas on cerebellar involvement in movement. *PubMed*, 11(2):457–487, June 2012. (Cited on pages 11 and 12)

[39] Louise H. Marshall and Horace W. Magoun. *Discoveries in the Human Brain*. 1998. (Cited on pages 10 and 11)

[40] S. Masoli and E. D'Angelo. Synaptic activation of a detailed purkinje cell model predicts voltage-dependent control of burst-pause responses in active dendrites. *Frontiers in Cellular Neuroscience*, 11:278, 2017. (Cited on page 14)

[41] S. Masoli, A. Ottaviani, S. Casali, and E. D'Angelo. Cerebellar golgi cell models predict dendritic processing and mechanisms of synaptic plasticity. *PLoS Computational Biology*, 16(12):e1007937, December 2020. (Cited on page 14)

[42] S. Masoli, D. Sanchez-Ponce, N. Vrieler, M. F. Rizza, and E. D'Angelo. Human purkinje cells outperform mouse purkinje cells in dendritic complexity and computational capacity. *Communications Biology*, 7:5, 2024. (Cited on page 31)

[43] Stefano Masoli, Sergio Solinas, and Egidio D'Angelo. Action potential processing in a detailed purkinje cell model reveals a critical role for axonal compartmentalization. *Frontiers in Cellular Neuroscience*, Volume 9 - 2015, 2015. (Cited on page 31)

[44] Marvin Minsky, editor. *Semantic Information Processing*. The MIT Press, Cambridge, Mass., 1968. (Cited on page 5)

[45] Tsugumichi Miyasho, Hiroshi Takagi, Hideo Suzuki, Shigeo Watanabe, Masashi Inoue, Yoshihisa Kudo, and Hiroyoshi Miyakawa. Low-threshold potassium channels and a low-threshold calcium channel regulate ca2+ spike firing in the dendrites of cerebellar purkinje neurons: a modeling study. *Brain Research*, 891(1):106–115, 2001. (Cited on page 46)

[46] ModelDB. Modeldb: A database of computational neuroscience models. `https://modeldb.science/`, 2025. Accessed: 2025-05-06. (Cited on page 31)

[47] National Center for Biotechnology Information (NCBI). Mapk8ip2 mitogen-activated protein kinase 8 interacting protein 2 [homo sapiens]. `https://www.ncbi.nlm.nih.gov/gene/23542`, 2025. (Cited on page 54)

[48] National Centre for the Replacement, Refinement and Reduction of Animals in Research (NC3Rs). The 3rs. `https://nc3rs.org.uk/who-we-are/3rs`, 2025. (Cited on page 55)

[49] Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99(4-5):335–347, 2008. (Cited on page 24)

[50] NEST Initiative. Neat: Neural ensemble analysis toolkit. `https://github.com/nest/NEAT`, 2025. (Cited on pages 29 and 53)

[51] NeuroMorpho.Org. Neuromorpho.org: A central resource for neuronal morphologies. `http://cng.gmu.edu:8080/neuroMorpho/index.jsp`, 2025. Accessed: 2025-05-06. (Cited on page 30)

[52] Nils J. Nilsson. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. Cambridge University Press, Cambridge, UK, 2010. (Cited on page 5)

[53] C. Pozzorini, S. Mensi, O. Hagens, R. Naud, C. Koch, and W. Gerstner. Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLOS Computational Biology*, 11(6):e1004275, 2015. (Cited on page 24)

[54] Jose Mario Prati, Andre Pontes-Silva, and Anna Carolyna Lepesteur Gianlorenco. The cerebellum and its connections to other brain structures involved in motor and non-motor functions: A comprehensive review. *Behavioural Brain Research*, 465:114933, 2024. (Cited on pages 9 and 12)

[55] M. F. Rizza, F. Locatelli, S. Masoli, D. Sanchez-Ponce, A. Munoz, F. Prestori, and E. D'Angelo. Stellate cell computational modeling predicts signal filtering in the molecular layer circuit of cerebellum. *Scientific Reports*, 11(1):3873, February 2021. (Cited on pages 14 and 29)

[56] Christian Rossert, Christian Pozzorini, Giorgio Chindemi, Andrew P. Davison, Cengiz Eroe, James King, Taylor H. Newton, Martin Nolte, Srikanth Ramaswamy, Michael W. Reimann, Felix Schurmann, Henry Markram, and Eilif Muller. Automated point-neuron simplification of data-driven microcircuit models. *arXiv preprint arXiv:1604.00087*, 2017. (Cited on page 24)

[57] S. Rudolph, A. Badura, S. Lutzu, S. S. Pathak, A. Thieme, J. L. Verpeut, M. J. Wagner, Y. M. Yang, and D. Fioravante. Cognitive-affective functions of the cerebellum. *Journal of Neuroscience*, 43(45):7554–7564, November 2023. (Cited on page 13)

[58] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 4 edition, 2020. (Cited on page 5)

[59] J. D. Schmahmann and J. C. Sherman. The cerebellar cognitive affective syndrome. *Brain*, 121(Pt 4):561–579, April 1998. (Cited on page 13)

[60] Mari Sepp, Kevin Leiss, Florent Murat, Konstantin Okonechnikov, Piyush Joshi, Evgeny Leushkin, Lisa Spanig, Noe Mbengue, Celine Schneider, Julia Schmidt, Nils Trost, Maria Schauer, Philipp Khaitovich, Steven Lisgo, Miklos Palkovits, Peter Giere, Lena M. Kutscher, Simon Anders, Margarida Cardoso-Moreira, Ioannis Sarropoulos snd Stefan M. Pfister, and Henrik Kaessmann. Cellular development and evolution of the mammalian cerebellum. *Nature*, 625:788â796, 2024. (Cited on page 8)

[61] T. Soda, L. Mapelli, F. Locatelli, L. Botta, M. Goldfarb, F. Prestori, and E. D'Angelo. Hyperexcitability and hyperplasticity disrupt cerebellar signal transfer in the ib2 ko mouse model of autism. *Journal of Neuroscience*, 39(13):2383–2397, 2019. Erratum in: J Neurosci. 2019 Aug 28;39(35):7029. doi: 10.1523/JNEUROSCI.1459-19.2019. (Cited on page 55)

[62] Sergio Solinas, Thierry Nieus, and Egidio DâAngelo. A realistic large-scale model of the cerebellum granular layer predicts circuit spatio-temporal filtering properties. *Frontiers in Cellular Neuroscience*, 4, 2010. (Cited on page 14)

[63] Roostaei T, Nazeri A, Sahraian MA, and Minagar A. he human cerebellum: a review of physiologic neuroanatomy. *PubMed*, 2014. Epub 2014 Oct 24. (Cited on page 8)

[64] The NEST Initiative. Nest simulator, 2025. (Cited on page 21)

[65] Michael S. C. Thomas, Robert Davis, Annette Karmiloff-Smith, Victoria C. P. Knowland, and Tony Charman. The over-pruning hypothesis of autism. *Developmental Science*, 19(2):284–305, 2016. Epub 2015 Apr 6. (Cited on page 53)

[66] F. Van Overwalle. Social and emotional learning in the cerebellum. *Nature Reviews Neuroscience*, 25(12):776–791, December 2024. Epub 2024 Oct 21. (Cited on page 13)

[67] F. Van Overwalle, M. Manto, Z. Cattaneo, S. Clausi, C. Ferrari, J. D. E. Gabrieli, X. Guell, E. Heleven, M. Lupo, Q. Ma, M. Michelutti, G. Olivito, M. Pu, L. C. Rice, J. D. Schmahmann, L. Siciliano, A. A. Sokolov, C. J. Stoodley, K. van Dun, L. Vandervert, and M. Leggio. Consensus paper: Cerebellum and social cognition. *PubMed*, 19(6):833–868, December 2020. (Cited on page 13)

[68] L. Vandervert. The prominent role of the cerebellum in the learning, origin and advancement of culture. *BMC*, 3:10, 2016. (Cited on page 13)

[69] Willem Wybo and contributors. Neat documentation, 2024. (Cited on pages 22, 27, 28, 29, and 57)

[70] Willem AM Wybo, Jakob Jordan, Benjamin Ellenberger, and Benjamin Torben-Nielsen. Data-driven reduction of dendritic morphologies with preserved dendrosomatic responses. *eLife*, 10:e60936, 2021. (Cited on page 22)

[71] Willem AM Wybo, Klaus M Stiefel, and Benjamin Torben-Nielsen. Electrical compartmentalization in neurons. *Cell Reports*, 26(7):1759–1773, 2019. (Cited on page 22)

[72] Chris I De Zeeuw, Stephen G Lisberger, and Jennifer L Raymond. Diversity and dynamism in the cerebellum. *PubMed*, 2021. Epub 2020 Dec 7. (Cited on page 10)

Acknowledgements

I would like to thank the laboratory of Brain and Behavioral Sciences of the University of Pavia for the help provided, for the time spent together on the project and also for the nice conversations and the good times we had during the lunch breaks. A special thanks to my supervisor Claudia Casellato for her availability and all the insightful tips provided, to my co-supervisor Martina Rizza for the extensive help and for always being very present during all the months of development, which was really appreciated and made me feel more secure in my work, and to my other co-supervisor Danilo Benozzo for suggesting the main idea of the thesis itself and the first steps of the work.
I also wish to thank Filippo Marchetti for helping me with the very troubled installation of NEST 3.7, Stefano Masoli for the help with the NEURON simulator and for contacting Willem Wybo about the process of installing and compiling the ionic channels, Dimitri Rodarie for showing me how to include better graphs to report BSB results and everyone that came to attend my informal seminar on automatic simplification tools.
I also deeply thank the Jülich Research Center, especially Willem Wybo and his team, for always being available in assisting me and clarifying my doubts.