Structural plasticity as a connectivity generation and optimization algorithm in neural networks

Sandra Diaz Pier

IAS Series
Band / Volume 47
ISBN 978-3-95806-577-2

JÜLICH
Forschungszentrum

Schriften des Forschungszentrums Jülich
IAS Series Band / Volume 47

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Structural plasticity as a connectivity generation and optimization algorithm in neural networks

Sandra Diaz Pier

Schriften des Forschungszentrums Jülich
IAS Series Band / Volume 47

ISSN 1868-8489 ISBN 978-3-95806-577-2

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Titelbild:	©Andrey – stock.adobe.com
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2021

Schriften des Forschungszentrums Jülich
IAS Series, Band / Volume 47

D 82 (Diss. RWTH Aachen University, 2021)

ISSN 1868-8489
ISBN 978-3-95806-577-2

Contents

Abstract

Our brains are formed by networks of neurons and other cells which receive, filter, store and process information and produce actions. The morphology of the neurons changes through time as well as the connections between them. For years the brain has been studied as a snapshot in time, but today we know that the way it structurally changes is strongly involved in learning, healing, and adaptation. The ensemble of structural changes that neural networks present through time is called structural plasticity. In this work, I present structural plasticity from its neurobiological foundations and the implementation of a model to describe generation and optimization of connectivity in spiking neural networks. I have targeted two relevant and open questions in the computational neuroscience community: how can we model biologically inspired structural changes in simulations of spiking neural networks and how can we use this model and its implementation to optimize brain connectivity to answer specific scientific questions related to healing, development, and learning. I present several studies which explain the implementation of structural plasticity in a well established neural network simulator and its application on different types of neural networks. In this thesis I have also defined the requirements and use cases for the co-development of tools to visualize and interact with the structural plasticity algorithm. Moreover, I present two scientific applications of the structural plasticity model in the clinical neuroscience and computer science fields. In conclusion, my thesis provides the basis of a software framework and a methodology to address complex neuroscience questions related to plasticity and the links between structure and function in the brain, with potential applications not only in neuroscience but also for machine learning and optimization.

Zuzamenfassung

Unsere Gehirne werden von Netzwerken aus Neuronen und anderen Zellen gebildet, die Informationen empfangen, filtern, speichern und verarbeiten und Aktionen ausführen. Die Morphologie der Neuronen verändert sich im Laufe der Zeit, ebenso wie die Verbindungen zwischen ihnen. Jahrelang wurde das Gehirn als Momentaufnahme in der Zeit untersucht, aber heute wissen wir, dass die Art und Weise, wie es sich strukturell verändert, stark am Lernen, Heilen und Anpassen beteiligt ist. Das Ensemble der strukturellen Veränderungen, die neuronale Netze im Laufe der Zeit darstellen, wird als strukturelle Plastizität bezeichnet. In dieser Arbeit stelle ich die strukturelle Plastizität von ihren neurobiologischen Grundlagen her und die Umsetzung eines Modells vor, das die Erzeugung und Optimierung der Konnektivität in neuronalen Netzen mit Spikes beschreibt. Ich habe mich auf zwei relevante und offene Fragen in der Gemeinschaft der Computational Neuroscience konzentriert: Wie können wir biologisch inspirierte strukturelle Veränderungen in Simulationen von neuronalen Netzwerken mit Spitzenbildung modellieren und wie können wir dieses Modell und seine Umsetzung zur Optimierung der Konnektivität des Gehirns nutzen, um spezifische wissenschaftliche Fragen im Zusammenhang mit Heilung, Entwicklung und Lernen zu beantworten. Ich stelle mehrere Studien vor, die die Implementierung der strukturellen Plastizität in einem gut etablierten Simulator für neuronale Netze und ihre Anwendung auf verschiedene Arten von neuronalen Netzen erklären. In dieser Arbeit habe ich auch die Anforderungen und Anwendungsfälle für die gemeinsame Entwicklung von Werkzeugen zur Visualisierung und Interaktion mit dem Algorithmus der strukturellen Plastizität definiert. Darüber hinaus stelle ich zwei wissenschaftliche Anwendungen des Strukturplastizitätsmodells in den Bereichen der klinischen Neurowissenschaften und der Informatik vor. Zusammenfassend kann ich sagen, dass meine Dissertation die Grundlage für ein Software-Framework und eine Methodik zur Behandlung komplexer neurowissenschaftlicher Fragen im Zusammenhang mit der Plastizität und den Verbindungen zwischen Struktur und Funktion im Gehirn bildet, mit möglichen Anwendungen nicht nur in den Neurowissenschaften, sondern auch für maschinelles Lernen und Optimierung.

Acknowledgements

First of all I would like to thank Prof. Abigail Morrison and Dr. Alexander Peyser for their mentorship, guidance, and friendship. In 2014 Prof. Abigail Morrison and Dr. Boris Orth gave me the opportunity to join the SDL Neuroscience where I have had the privilege and joy of working in the intersection between neuroscience and HPC and do my PhD, for which I will always be grateful to them. I wish to thank Prof. Uwe Naumann for reviewing my thesis and for his valuable comments, and Prof. Gerhard Lakemeyer and Prof. Martin Grohe for being part of my doctoral examination committee. I would also like to thank the co-authors in the different papers which are part of the research presented in this dissertation, in particular Prof. Thanos Manos, Prof. Peter Tass, Dr. Christian Nowke, Dr. Butz-Ostendorf, and Dr. Mikael Naveau. I want to express my deep appreciation to all the collaborators and the users of the structural plasticity tools with whom I have walked this research path. I would like to thank my friends and colleagues from the SDL Neuroscience who have provided me their support and helped me complete this step in my career. Finally, I would like to thank my family for their endless support and specially to my dear son Aaren and my wonderful husband Emanuele who have always given me the strength and motivation to give the best of me and achieve my dreams.

This work was supported by the Helmholtz Association through the Helmholtz Portfolio Theme "Supercomputing and Modeling for the Human Brain" and its Initiative and Networking Fund as well as by the Jülich Aachen Research Alliance (JARA). This research was also supported by the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2).
I gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC) and JARA on the supercomputer JURECA at Forschungszentrum Jülich.

Eidesstattliche Erklärung

Eidesstattliche Erklärung
Ich, Sandra Diaz Pier, erkläre hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind un selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden. Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand und Wissentschafliche mitarbeiter des SimLab Neuroscience des Forschungszentrum Jülich angefertigt;

2. Sofern irgendein Bestandteil diseser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;

3. Wenn immer andere eigene- der Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;

4. Wenn aus anderen eigenen-oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben.

5. Diese Dissertation is vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;

6. Alle wesentlichen Quellen von Unterstützung wurden benannt;

7. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;

8. Teile dieser Arbeit wurden zuvor veröffentlicht, ersichtlich im Abschnitt *Publications and contributions*

Publications and contributions

Journal publications and contributions

- **Automatic generation of connectivity for large-scale neuronal network models through structural plasticity** Sandra Diaz-Pier, Mikael Naveau, Markus Butz-Ostendorf, and Abigail Morrison, 2016. Frontiers in neuroanatomy, 10, p.57.

  https://doi.org/10.3389/fnana.2016.00057. (citations: 29)

  Chapter *Implementation and usage of a structural plasticity algorithm in NEST* is based on this publication.

  Contributions

  Under the supervision of Abigail Morrison and Markus Butz-Ostendorf, the author together with Mikael Naveau implemented and integrated the code. Mikael Naveau implemented the infrastructure for synaptic element growth. The author focused on the synapse deletion, the interfaces between simulation and user scripts, refinement of the synaptic element information exchange in agreement with the MPI and multithreading parallelization strategy of the simulator, documentation, and integration with the code base of NEST. The author also performed the simulations, scaling tests, and conducted the data analysis. All co-authors participated in writing the manuscript.

- **Toward rigorous parameterization of underconstrained neural network models through interactive visualization and steering of connectivity generation** Christian Nowke, Sandra Diaz Pier, Benjamin Weyers, Bernd Hentschel, Abigail Morrison, Torsten W. Kuhlen, and Alexander Peyser, 2018. Frontiers in neuroinformatics, 12, p.32.

  https://doi.org/10.3389/fninf.2018.00032. (citations: 7)

  Chapter *Steering and interactive visualization of structural plasticity* is based on this publication. This publication has also been used in the dissertation of Christian Nowke.

Contributions Under the supervision of Torsten Kuhlen, Bernd Hentschel and Benjamin Weyers, Christian Nowke implemented the nett framework and the infrastructure to support structural plasticity visualizations. The author and Christian Nowke implemented the scripts which link simulation and visualization. Abigail Morrison, Alexander Peyser, and the author defined the use cases and the functional requirements. Under the supervision of Alexander Peyser and Abigail Morrison, the author performed the simulations, scaling tests, user manuals, documentation, and conducted data analysis. The author performed the generalization of the scripts to gather information from different network models. The author, Christian Nowke, Abigail Morrison, Alexander Peyser and Benjamin Weyers wrote the manuscript.

- **Long-term desynchronization by coordinated reset stimulation in a neural network model with synaptic and structural plasticity** Thanos Manos*, Sandra Diaz-Pier and Peter A. Tass, 2021. Frontiers in Physiology.

  doi: 10.3389/fphys.2021.716556. (Provisionally accepted)

  Chapter *Clinical applications* is based on this publication.

  Contributions Thanos Manos and Peter Tass conceptualized the brain model and the stimulation protocols based on previous work and publications. Thanos Manos implemented the model in NEST 2.18. The author integrated structural plasticity to the model and implemented the code to enable structural and synaptic plasticity to work together in NEST. The author performed the simulations, the parameter exploration, and conducted data analysis. All authors analyzed the results and wrote the manuscript.

Other journal papers not directly covered in this dissertation

- Blundell, Inga, et al. ”Code generation in computational neuroscience: a review of tools and techniques.” Frontiers in neuroinformatics 12 (2018): 68.

- Meysam Hashemi, Ph.D.; Anirudh N Vattikonda; Viktor Sip; Sandra Diaz-Pier; Alexander Peyser; Maxime Guye; Fabrice Bartolomei; Marmaduke M Woodman; Viktor K. Jirsa. ”On the influence of prior information evaluated by fully Bayesian criteria in a personalized virtual brain model of epilesy spread”. submitted to PLOS Computational Biology.

1 Introduction

1.1 Introduction and Motivation

Models of large scale neural networks are an important tool for understanding the mechanics of the brain [64, 47, 32]. Such models are created based on experimental information that has been collected for years by neuroscientists and combine mathematical methods with algorithms to reproduce observed behavior. The connectome – the collection of all connections between neurons in the brain – gets its basic structure during brain development and is continuously refined during the lifespan of a living being. It is known that the connectivity of the network plays a role in defining the way function is achieved at higher levels of activity. We know that the changes in synaptic connections in the brain are an essential component of learning, memory, adaptation to stimuli, and healing after lesions. Understanding how the connectome emerges and which mechanisms regulate its changes is fundamental for neuroscience to reach a better understanding of how brain structure and higher function relate.

Nevertheless, obtaining accurate measurements of connectivity is complex, even with the most advanced experimental techniques, due to the resolution of sensors and difficult access to the target areas. Non invasive techniques such as diffusion tensor imaging (DTI) and functional magnetic resonance imaging (fMRI) scans can provide a glimpse to the real complexity of the problem in structure and function. Higher resolution techniques like electron microscopy [60], photostimulation [31] and electrophysiological recordings [149] provide more detailed connectivity information of specific regions. Regardless, creating an exact connectivity map of even a small region of the brain is extremely challenging [51, 52, 34, 123]. This poses a significant problem for the modeling approach, as connectivity must be specified. For small networks, parameter scans can be carried out with respect to the unknown or imprecisely known connection probabilities between populations. For larger networks, which are more costly to simulate and also potentially have many more unknown connectivity parameters, this approach is hardly feasible.

With the emergence of new high performance computing technology in the last decade, the simulation of large scale neural networks has become possible. Thanks to this, the reproduction of the behaviour and structure of individual regions of the brain has become a feasible target for neuroscience in the next years. Due to the number of synaptic connections between neurons and the complexity of biological networks, most neural network models have manually defined or static connectivity. This connectivity is commonly obtained by combining experimental and statistical data. However, the ability to model the dynamic generation and deletion of the links among neurons, locally and between different regions of the brain, is crucial to study important mechanisms associated with learning, memory and healing. One way to address the issue of modeling connectivity within a neural network is to allow a network model to determine its own suitable connectivity to achieve target activity patterns, e.g. experimental measurements of the spiking frequency, which is easier to measure accurately than connectivity. In addition to addressing the problem of network model specification, a framework that accounts for the appearance and disappearance of synapses on the basis of network activity can provide insight into how connectivity is generated during development and learning or even on how healing after lesions takes place [33]. It can also help understand how certain structures arise as a result of exposition to adequate external stimuli during critical periods in the development of the brain [68] and the mechanisms underlying experience dependent structural synaptic plasticity [74].
For long time, simulations of neural networks have incorporated synaptic plasticity but few simulators are prepared for modeling structural plasticity. This is mainly due to the fact that creating and deleting synapses in a simulation can be computationally costly. Different models of structural plasticity have been suggested in the last years and different parameters can be used to guide the model, such as: changes in the intracellular calcium concentration of the neurons, average membrane potential, correlated activity among neurons, distance between the pre- and post- synaptic neuron, homeostasis, ionic gradients, among many others. Still, the lack of a reliable simulation platform which supports the modeling of structural changes through long periods of biological time hinders our current ability to understand this phenomenon.
Studying the dynamics of connections in simulations of spiking neurons is a challenging but critical endeavor, indispensable if we want to understand learning, how the brain develops, how it changes and adapts to the stimuli we perceive with our senses, and how we can design new ways to treat brain disorders which consider not only the anatomy and physiology but also the aftereffects of stimulation, surgery, and medicaments in the brain's function and structure. Computational neuroscience requires a reliable, efficient, and accessible platform to understand

plasticity in the brain. Today, inspired by these questions and supported by the computational power now available, it is possible to meet this challenge. My thesis focuses on the modeling of structural changes in neural networks and its implementation in a simulation framework. During my work, I have used and modeled structural plasticity in the interface between neuroscience and computer science. As many algorithms in computer science, the work I present here has been inspired by nature and artificially refined to fulfill practical tasks.
I have carried out my work in the context of the large international brain research initiatives including the Human Brain Project in Europe, the US BRAIN initiative and the China Brain Project among others. So far, structural plasticity has been insufficiently addressed by these large research initiatives and by smaller computational neuroscience projects. My work aims at making first steps into a simulation platform for better understanding the changes in the structure of the brain and their impact in its function.
The main questions explored in this thesis are:

1. **How can we model and use biologically inspired structural plasticity in large simulations of spiking neural networks?**

2. **How can we use structural plasticity as an algorithm to artificially optimize the connectivity within a neural network to answer specific scientific questions?**

1.2 Description of this thesis

In Chapter 2 I provide an introduction to structural plasticity in the brain. The chapter discusses the basic concepts of this phenomenon and its role in learning, healing and development. It also introduces a model to describe structural plasticity algorithmically.
The second application of structural plasticity shown in this thesis relates to machine learning. Chapter 3 provides a short introduction to optimization which is one of the fundamental steps in the learning process. This chapter describes machine learning algorithms as background for studying the role of structural plasticity in learning in spiking neural network models. The chapter also includes an overview of analytical frameworks which are useful to study models based on coupled differential equations and control theory.
Chapter 4 covers the implementation details of the model introduced in Chapter 2. I describe the integration of such a model into the NEST simulator, a well known neural network simulator with

a large and strong community and development structure. This chapter also presents example networks, performance analysis and benchmarking of the scalability of this implementation.
As mentioned before, the potential of the model that I use in this thesis goes beyond the simulation of neurobiological modifications in the structure of the network. The algorithm can also be used to automatically generate connectivity in a network which enables it to have a particular firing rate per population. However, the algorithm also has its limitations and it is sensitive to changes in its parameters. This is a frequent problem in models used in neuroscience and other disciplines. In Chapter 5 a sensitivity analysis on the parameters which define the structural plasticity algorithm is presented. A formalization of the model is made from the control theory perspective in order to identify the inputs, outputs, control elements and observables in such a model. I present an interactive steering and visualization tool which allows the modification of variables in the algorithm and observe the output of a NEST simulation while running.
The second half of the thesis is dedicated to applications of the structural plasticity algorithm for different use cases. Chapter 6 discusses the integration of structural plasticity in a model to study the effects of Coordinated Reset Therapy (CRT) in patients with Parkinson's disease. Using a model of the Sub Thalamic Nucleus (STN) and Globus Palidus externus(GPe) in combination with structural and synaptic plasticity, it was possible to define regimes of plasticity with which we could reproduce long lasting effects of CRT. The methodology and results of using different combinations of plasticity parameters in this model are shown and discussed.
Chapter 7 introduces structural plasticity as an artificial optimization rule to generate networks which exhibits simple behaviour. To do this, learning hyperparameter optimization techniques were integrated on top of the structural plasticity algorithm. With these experiments, it was possible to identify features in the parameters of structural plasticity which were linked to general optimization tasks. These tasks involved making the networks move towards stable firing rate regimes, changing firing rates depending on stimuli.
The last chapter of this thesis provides conclusions and discusses the future developments and applications of the structural plasticity framework. Techniques which can be used in the future to overcome its current limitations are quickly explored.

1.3 Contribution

The main contribution of this thesis is a framework to use structural plasticity as a technique to study neurobiological changes in simulations of neural networks of spiking neurons and as an artificial optimization algorithm for generative connectivity. I have built a software infrastructure for modeling, simulating, visualizing and analyzing structural plasticity to modify, generate and optimize connectivity in simulations of neural networks. I have also used this infrastructure to investigate specific scientific questions. I have simulated structural changes induced by Coordinated Reset Therapy for the treatment of Parkinson's disease and concluded that time lapses between stimulation play an important role in its therapeutic effects. I have also used structural plasticity to propose a way to study evolutionary tuning of learning mechanisms in the brain applicable to artificial intelligence. With this work, I show the potential of structural plasticity as a biological optimization algorithm applicable to neuroscience and learning in computational tasks.

2 Structural plasticity in the brain

In this chapter I provide an introduction to neural networks, brain structure and plasticity from the neuroscience and computational perspectives. Its goal is to provide a general overview of terms and concepts which are useful to understand the work I have performed in my thesis. I start with basic concepts about the structure and function of the brain considering its multiscale nature. Then I introduce structural plasticity identifying a set of characteristics which can be used to aid modeling of this phenomenon. Finally, I present the main brain simulation software used in my work.

2.1 The brain as a multiscale system

The complexity of understanding the brain starts from its multiscale nature. The brain has processes taking place at different spatial and time scales. The results and effects of these processes combine hierarchically to produce new structures and function from one scale to the next scale. From the molecular level to the cellular and network level all the way to the whole organ, the brain exhibits a rich interplay of multimodal (e.g. chemical, electrical, electromagnetical, and physical) dynamics. The time at which these dynamics take place also spans from milliseconds to years, which makes it hard to create unified mathematical models to describe them. In terms of simulation technology, this complexity has been addressed by developing independent tools focusing on specific spatial and time scales of study. This approach, restricted by computing capacity, limited the study of the interactions between scales making it hard to understand the relationships between functions and structures. The brain should be studied as a continuously changing multiscale system [46, 43, 42]. Because of this, it is essential to continue the refinement and implementation of computationally efficient tools which leverage new hardware infrastructure and provide new capacity to researchers, enabling them to model, simulate and study the brain in increasingly complex setups [7].

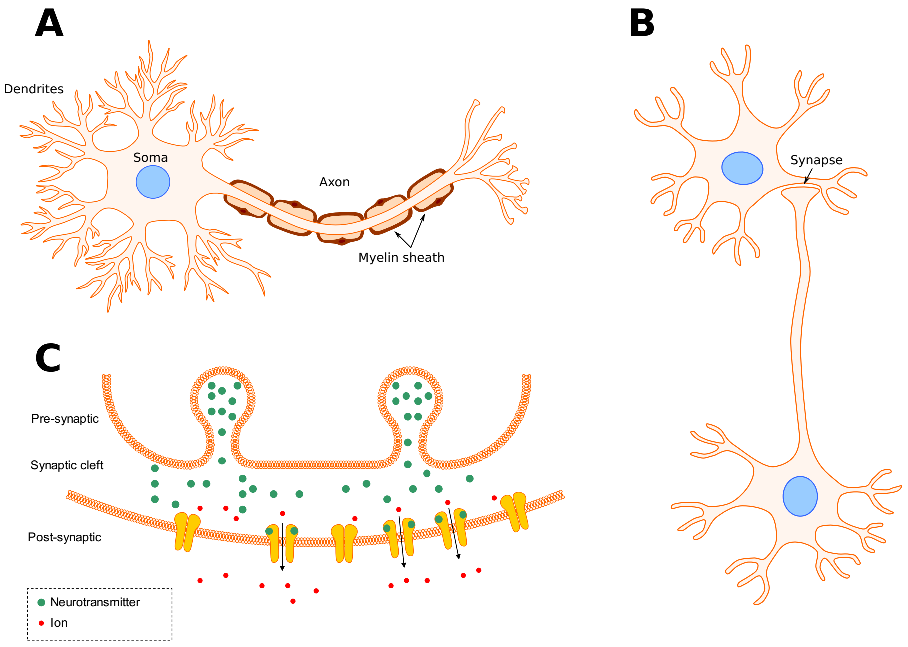


Figure 2.1: A) Schematic drawing of a neuron and the basic parts of most interest for this thesis work. B) Diagram of an axodendritic synapse connecting two neurons. C) Enlarged schematic of a synapse, where the pre-synaptic bouton, the synaptic cleft and the post synaptic spine are illustrated. Illustrations by Motifolio Inc.

2.2 Basic concepts about the structure and function of biological neural networks

At the cellular level the brain is a complex ensemble of processing units with various functional and morphological characteristics. Several types of neurons, glia and inter neurons have so far been identified by experimental neuroscientists. Neurons are the most functionally and structurally studied cells in the brain because it is thought that they perform most of the information processing. Each of these cells is a complex electrochemical machinery which grows and interacts with neighboring cells. A simplified diagram of a neuron can be seen in Figure 2.1.
The dendrites and the soma of a neuron are the two places where most of the input to a neuron is received. The shape of the ramifications that the dendrites form can be used as a way to

morphologically categorize neurons. The axon of a neuron, on the other hand, serves mostly as means to sending signals to other neurons. The axon is an elongation of the membrane from the soma, which is usually longer and thinner than the dendrites. In most vertebrates, the axons are covered by a myelin shed which allows rapid conduction of electrical signals.

Neurons exhibit electrical accumulation and transmission properties which depend on the exchange and difference of ionic concentrations between the inner and the outer space of the cell. In the early stages of neuroscience, it was believed that the brain was formed by a unique continuous network and this gave birth to the reticular theory. Today we know that neurons are independent processing units but they are still connected to each other mostly by means of synapses [164]. This allows the generation of networks of neurons. From experiments, we know that neurons in the cortex have approximately $10,000$ connections to other neurons. We also know that connections are more frequent between neighboring neurons.

Action potentials or spikes are produced when the membrane potential of a neuron surpasses a given threshold, triggering a chain reaction of different ion flux through the membrane along the axon. The membrane potential of the neuron changes depending on its interaction with its surroundings and on the electrical signals that it receives through its synapses.

Synapses are established between two contact points, one in the pre-synaptic neuron (the neuron sending the electrical signal) and another one in the post-synaptic neuron (the neuron receiving the information). Most synapses are formed as a connection between the axon of the pre-synaptic neuron and a dendrite or soma of the post-synaptic neuron. The actual place where the synapse occurs is where an axonal bouton and a spine meet. Figure 2.1 in segments B and C show what a synapse looks like. There is actually no contact between the two parts; they stay separated by what is called the synaptic cleft. The bioelectrical mechanism that propagates an action potential flow on the axon of a neuron into a graded potential on the dendrite of another can deffer according to the type of neurotransmitter that the synapse works with. Described generally, synapses can have an excitatory or inhibitory nature, contributing to the increase or decrease of the membrane potential in the post-synaptic cell. Synapses can vary their strength, thus regulating the effect of the pre-synaptic neuron's activation on the post-synaptic neuron.

Neural networks are adaptive, morphologically changing assembles of neurons which receive, process and generate information as electrical signals [138]. Even though the large scale structures of the brain are well known and understood, we still have little information about the detailed organization of the whole brain at cellular level. Using the latest imaging and measuring techniques (EEG, fMRI, 2 Photon Imaging, Array Tomography, etc) we start now to identify connectivity maps between neurons, within groups of neurons and among regions in the brain. So far, a full, consistent, complete and reliable map of the connectivity in the human brain does not exist. Moreover, since the earliest studies of the brain, it has been noted that the connectivity in the brain changes through time.

2.3 Structural plasticity

Changes in the morphology of the neurons and the connections between them is what we know as structural plasticity. When using this term in the context of this thesis, I refer particularly to the generation and deletion of synapses between neurons. Structural plasticity plays a main role in brain development, learning, adaptation, healing and other brain processes. Kehayas and Holtmaat [85] give an excellent introduction to structural plasticity and its role in learning, memory, rewiring following enriched experience after sensory deprivation or stimulation, and its relationship with long term functional synaptic plasticity.

During brain development, neurogenesis takes place and the basic connectivity between neurons is established. Neurogenesis is the final step in which neural stem cells divide and specialize to become new born neurons. Newly born neurons will migrate to their final location and, once there, they will mature, grow and create connections with other cells. Neurons tend to create synapses with other neurons in their vicinity and with which they establish affinity by following biochemical attractors. They also seem to follow rules to reach and stabilize a certain level of electrical activity and intracellular ionic concentrations. In this work I will not discuss neurogenesis but I will show how structural plasticity can be used as a generative algorithm to build connectivity in a totally disconnected network using homeostasis. Homeostasis is the dynamic preservation of equilibrium, and, in the context of structural plasticity, the term is used to indicate the process that neurons use to achieve a metabolic state by changing their morphology and connections. The homeostatic process implies that the neuron has a cell-autonomous set point, a level of activity to reach and hold. This set point was first proposed in Maffei and Turrigiano [102] as a synaptic scaling factor and later experimentally demonstrated in Hengen et al. [67] and Hengen et al. [66] as cell-autonomous.

Even though the period of highest plasticity in the brain is during development, the mature brain retains the ability to change its structure. One of the experimental areas which provides strong support to structural changes in the mature brain is related to the study of experience-dependent plasticity. For example, dendritic spines have been identified to be very changing and moving

elements [152, 75]. Experiments have shown that only about one half of the spines in the barrel cortex of the adult mouse will persist for a period longer than a month [152]. Several studies illustrate how stimuli deprivation has a strong impact in the formation of neural structures and reorganization of the sensory cortex [82, 84, 83, 91].
Changes in connectivity during healing in the brain is also a thoroughly studied topic in clinical medicine. Rewiring of neural networks plays an essential role for the compensation of missing or distorted input for processing capabilities in the network. Since the 1960s, studies on experience dependent structural plasticity have shown that stimulus deprivation triggers rewiring of the networks, leading to synaptic loss and rewiring of connectivity to neighboring areas [76]. For example, in [162] Wiesel and Hubel discuss the ability of the network to recover functionality after deprivation of stimuli in early age. They showed that there are periods of time in early development during which critical steps in the formation of robust connectivity take place; permanent damage is produced when these periods of plasticity are hindered. Their results show, for example, that young cats did not achieve any degree of recovery if they were exposed to eye stimuli deprivation for the first 3 months of life. Similar studies also support the hypothesis that plasticity in the brain is strongest during early life and that critical periods of synaptic plasticity are essential to avoid permanent loss of certain functions. Although still highly contentious, the adult brain has been shown to be still plastic and able to rewire to reestablish certain functionality after injuries or deprivation [106] or a pathologic state is reached. For example, structural plasticity has been proposed as an underlying mechanism in deep brain stimulation therapies targeted to reduce symptoms of Parkinson's and other related diseases [62]. Structural plasticity has also been studied in the context of stroke recovery. In Gerrow and Brown [57] the authors discuss how after stroke, the peripheral areas to the affected stroke zone, also called peri-infarct zones, show expression of growth factors similar to those found in the brain during development. Gene and protein expression is altered in these zones, contributing to the rewiring of the neural network around the stroke zone. Changes in the connectivity involve both intracortical and long range projections. Evidence confirms that new connections form mainly in the regions functionally related to the affected zone.
Another relevant field for structural plasticity is learning and adaptation. Donald O. Hebb, the father of the current theory about learning, highlighted in his work several mechanisms at cellular level which are linked to this effect [63]. Among these mechanisms, he suggested that the correlated activation of neurons led to the enhancement in the strength of synapses among those neurons, leaving a long lasting trace which could be later translated into learning. He stated

that the basis for this mechanisms was the formation of new contacts between neurons and the strengthening of existing ones.
Structural plasticity plays a key role in learning by removing weak synapses and enabling new contacts with other neurons which, if reinforced by correlated activation, can lead to learning a new function. This has been the topic of several publications in the last years [41, 148, 50]. The combination of synaptic and structural plasticity leads to a powerful neurobiological infrastructure to support the acquisition of new functions and refinement of existing ones.

2.3.1 A model of structural plasticity

A theoretical model of structural plasticity that incorporates the dynamic generation, deletion and rewiring of synapses within a network was presented by Butz and van Ooyen [22]. In this model, synapses are represented as connections between pre and a post synaptic elements. The growth or recession of these synaptic elements is an independent process for each neuron. The model is based on the idea that plasticity in cortical networks is mainly driven by the need of individual neurons to homeostatically maintain their average electrical activity. As a consequence, if activity is lower than a desired set-point, neurons will form synaptic elements, and remove them when activity becomes too high. Additionally, a minimum level of activity is needed to form synaptic elements at all. If activity falls below this level the neuron will remove synaptic elements, too. Results show that small networks of hundreds or thousands of neurons robustly grow towards a stable homeostatic equilibrium of activity and connectivity. An important advance on earlier work is that all cell types have different desired average firing rates (achieved by different homeostatic set-points) and develop connectivity accordingly. These local rules for structural plasticity were shown to account for network rewiring after a partial loss of external input (deafferentation) and shows remarkable similarities with biological data from network rewiring in the primary visual cortex after focal retinal lesions [82, 165]. Further analysis by Butz et al. [23] of changes in network topology revealed that betweenness centrality could be used as an indicator of successful brain repair, in the sense that it is related to the ability of the neurons to restore their electrical activity by network rewiring. The authors concluded that structural plasticity may account for network reorganization on different spatial scales.
The original formulation of the structural plasticity algorithm defined in Butz and van Ooyen [22] consists of three repeating steps which are described as follows:

1. Update in electrical activity and intracellular calcium concentration. The electrical activity

is calculated for each neuron on a millisecond timescale. The time-averaged level of the neuron's electrical activity drives changes in neuronal morphology. Intracellular calcium concentration is updated according to the electrical activity as follows:

$$\frac{d\mathrm{Ca}}{dt} = \begin{cases} -\frac{Ca(t)}{\tau} + \beta & \text{if the neuron fires} \\ -\frac{Ca(t)}{\tau} & \text{otherwise} \end{cases} \tag{2.1}$$

where τ is the calcium decay constant and β is the calcium intake constant which defines how the intracellular calcium concentration increases each time the neuron fires. Calcium concentration is linearly proportional to average firing rate and thus is the measure that is used to guide the growth dynamics of the synaptic elements.

2. Update in synaptic elements. The detailed morphology of the synaptic elements is abstracted, and is represented in this formulation only by the number of possible synaptic contacts on axons (axonal elements representing axonal boutons: senders of synaptic activity) and on dendrites (dendritic elements representing dendritic spines: receivers of synaptic activity) collectively called synaptic elements. Synaptic elements are created or deleted according to a homeostatic rule. In general, the homeostatic rule will create synaptic elements when the activity is lower than the desired set point and delete them when the activity is higher until the desired activity level is achieved. This homeostasis is represented by a function which defines how quickly new elements are created or deleted according to the current level of electrical activity. The original work from Butz and van Ooyen considers two types of growth function, linear and Gaussian:
Linear:

$$\frac{dz}{dt} = \nu(1 - \frac{1}{\varepsilon}Ca(t))$$

where ν is the growth rate and ε is the target level of calcium concentration that the neuron should achieve. An example of such a growth rule can be seen in Figure 2.3 a.

Gaussian:

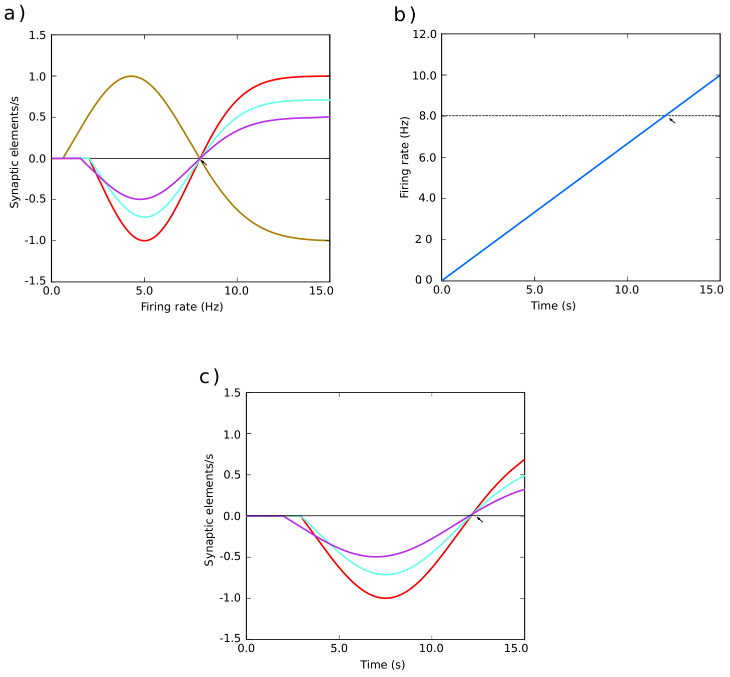


Figure 2.2: Growth rate curves and their effect on firing rate and synaptic element creation / deletion. a) Example of growth rate curves determining the rate of creation or deletion of synaptic elements in the structural plasticity model. The parameters which define the shape of the curve are two firing rates, the minimal firing rate for creating/deleting synaptic elements η and the target firing rate ε, and the growth rate ν which is the value of the curve in synaptic elements/s when the firing rate $\lambda = (\varepsilon - \eta)/2$. The red, cyan and purple curves have a negative value of ν which implies that synaptic elements will be deleted when the current firing rate is less than the target rate. These curves are therefore suitable for inhibitory synapses. Conversely, synaptic elements will be created when the current firing rate exceeds the target. The brown curve has a positive ν which works in the opposite way. All curves display different values of η; in particular, the cyan curve has a negative value of η. In these cases, all curves have a target firing rate ε of 8 Hz. It is important to note the slope of each curve close to the target firing rate ε; this slope is critical for the stability of the optimization algorithm. b) Firing rate externally imposed on sample systems with Gaussian growth curves shown in (a); and c) the resulting evolution of synaptic growth rate through time due to the firing rate changes depicted in (b). See Figure 5.3b for an equivalent Gaussian growth curve for the two-population example in this chapter, and the resulting free (not driven) dynamics in Figure 5.3c, d, and e.

A Gaussian curve (Figure 2.3 b) is an example of a homeostatic rule describing the growth rate of connection points for neurons. The parameters defining the growth and decay of synapses are the minimum firing rate η required to generate synaptic elements (or destroy

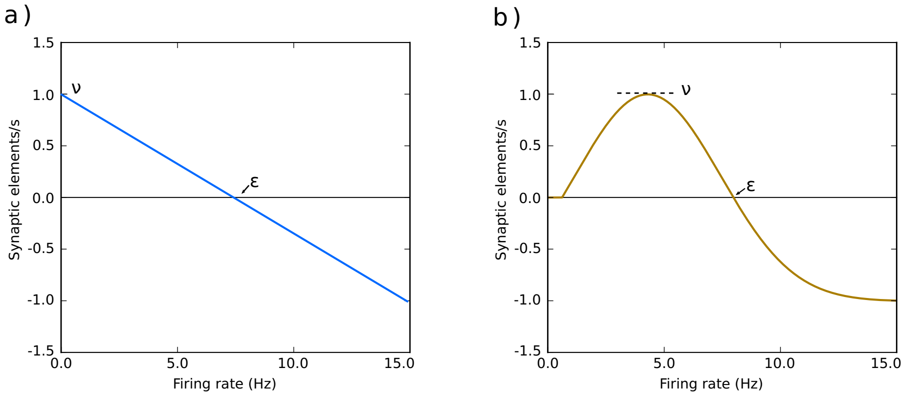


Figure 2.3: a) Example of a linear growth function. b) Example of a Gaussian growth function.

them, depending on sign of ν), the value ν of the growth rate curve when the firing rate is $(\varepsilon - \eta)/2$, and the target firing rate ε. Modifying these values alters the way connectivity is created and destroyed in the network.

Thus, to calculate the number of synaptic elements per second ($\mathrm{d}z/\mathrm{d}t$) to create (or remove, if negative), we use:

$$\frac{\mathrm{d}n}{\mathrm{d}t} = \nu\ \mathrm{H}[\lambda - \eta] \left[2\ \mathrm{pow}_2 \left(- \left[2\frac{\lambda - \eta}{\varepsilon - \eta} - 1 \right]^2 \right) - 1 \right] \tag{2.2}$$

where $\mathrm{pow}_2\, x$ is the power function 2^x and $\mathrm{H}[x]$ is the Heaviside step function equal to 0 when $x < 0$, otherwise 1. Equation (2.2) is equivalent to the Gaussian used in Diaz-Pier et al. [37] and in the original model by Butz and van Ooyen after directly replacing the calcium concentration with the firing rate λ.

Figure 2.2 shows an example of the changes in the synaptic growth rate (panel c) using different gaussian growth rates (panel a) as the firing rate of a model neuron changes through time (panel b).

A synaptic element is formed (or deleted) when the rounded down z value increases (or decreases) by one. Newly-formed synaptic elements are initially vacant and available for synapse formation.

3. Update in connectivity. In every connectivity update, available synaptic elements allow the formation of new synapses, while deleted synaptic elements dictate synapse removal. Every available synaptic element has the same probability to be randomly chosen for a new connection. Synaptic elements to be deleted are also chosen in a uniformly random manner from the pool of already connected elements. It is important to notice that in this algorithm, when a synapse breaks due to the deletion of one synaptic element, the counterpart remains and becomes vacant again. This remaining counterpart can form a new synapse at the next update in connectivity. This effect models network rewiring by re-routing of axons or dendrites, a process known as reactive synaptogenesis [87].

An important characteristic of this algorithm is that it relies on global communication to update the connectivity in the network, since available compatible synaptic elements must be matched during the execution of the algorithm to create new connections. This must be taken into consideration for the design of any implementation of this model.

2.3.2 Essential features of structural plasticity

The following list summarizes the main abstract features of structural plasticity:

1. Structural plasticity considers the creation or deletion of synapses between a pre- and a post-synaptic neuron.
2. It is a slow process as compared to synaptic plasticity and neural dynamics.
3. It plays a relevant role in development, healing, learning and re-learning.
4. The neuron has a local view of its metabolic needs and is guided by homeostasis. It works as a controller of the neuron's metabolism.
5. Changes in the activity of one neuron have an impact on all other neurons connected to it. This impact might be subtle or strong depending on the strength of the synapses they share.
6. Neurons connect with a higher probability to other neurons closer to them.
7. Unused connections become weaker and are subject to be removed in favor of new connections which might be more useful to the neuron.

Please see Chapter 8 for a discussion on how this thesis implements the aforementioned features.

Scale	*Example simulators*
Whole brain	TVB [131]
Networks of neurons	NEST [59], Arbor [4], Neuron [25], Brian [140], GeNN [166]
Morphologically detailed neurons	Arbor, Neuron
Ion channels and molecules	STEPS [70], GROMACS [71]

Table 2.1: Examples of simulators for the different scales of the brain.

2.4 Simulation of neural networks

Simulations constitute today one of the main tools that neuoscientists have to understand how the brain works. The dynamics of the diverse functional structures in the brain can be mathematically modeled to answer specific scientific questions. The smaller the spatial scale, the larger the computational complexity and the smaller the time scale where relevant dynamics are found. Several simulators for different spatial scales of the brain exist and are supported by the computational neuroscience community. Examples of them can be seen in Table (2.1).
Structural plasticity takes place in the connections between neurons and, for this reason, in this thesis I concentrate on the scale of networks of neurons. In agreement to the chosen scale, the main simulation platform I have worked on is NEST [59]. NEST is a simulator of networks of spiking neurons where the morphology is abstracted and neurons are treated as points without explicit axons or dendrites. Networks can be created by connecting neurons using different types of synapses. In NEST, neurons are modeled by sets of differential equations which describe the changes in their membrane potential. Synapses are also modeled using differential equations which describe how action potentials are propagated from the pre- to the post-synaptic neuron.
NEST has a large community of users and its main target applications are information processing, network activity dynamics, learning and plasticity. The NEST core is written in C++, with an interface in a domain language called SLI and a higher level user interface in Python. NEST can be run in laptops or supercomputers, and ensures the exact reproducibility of simulation results on any platform used. The simulation scales that NEST supports go from one neuron to several millions of them. Additionally, NEST is parallelized using MPI and openMP, which allows the simulation of large scale networks. The largest simulation performed to date in the K supercomputer in Japan included $1.86x10^9$ neurons and $11.1x10^12$ synapses, taking $793.42s$ to build the network and $2481.66s$ to simulate $1s$ of biological time [94].
Another relevant simulator in the context of this thesis is The Virtual Brain (TVB) [131]. TVB is

a simulator of neural mass models which describe the average activity of large groups of neurons. This makes TVB a simulator for a larger spatial scale than NEST. TVB allows the simulation of whole brain signals at a level which can be compared to experimental data obtained by MRI, fMRI or EEG. Typical simulations with TVB can consider biological times from milliseconds to minutes or hours. TVB is relevant in the context of structural plasticity because average firing rate of regions of the brain can be linked to components of homeostatic rules. Simulations done with TVB are faster than NEST because the level of complexity and detail is lower. Both simulators can be used together to make more efficient and useful predictions about the way structural changes at neuron level impact the function at network level.

2.5 Summary

In this chapter I introduced useful concepts about neuroanatomy, computational neuroscience and mathematical models to understand structural plasticity and its integration in simulations of networks of spiking neurons. I introduced structural plasticity as the key concept in this thesis and extracted a set of characteristics from its definition in the neuroscience domain. These characteristics will be used in the next chapters as requirements for biologically realistic modeling of this phenomenon and guide the implementation. This chapter also provides a brief context on current simulation technology used in computational neuroscience.

3 Introduction to optimization algorithms and machine learning

In the previous chapter I introduced several concepts about neurobiology and computational neuroscience which are useful to understand structural plasticity. In this chapter I shift the perspective to the computer science field, specially focusing on algorithmic optimization and machine learning. This shift is motivated by the fact that structural plasticity works as a mechanism to find and learn functionally optimal network structures which support specific homeostatic conditions. The history of computer science has seen the emergence of a variety of algorithms inspired by nature which are useful to solve optimization problems. The goal of this chapter is to provide a quick introduction to algorithms commonly used in machine learning as a background to understand structural plasticity as an optimization algorithm. This chapter also introduces useful formalizations which will later help understand the steering and visualization tools developed for the analysis of structural plasticity in simulations. With this background it will be easier to explain the role of structural plasticity in the optimization of connectivity in neural networks, but also its potential to solve problems outside of the neuroscience realm.

3.1 Algorithmic optimization

Swarms In 1989, Craig Reynolds [124] developed the concept of swarm intelligence. Swarms are based on the movement of animal ensembles such as flocks of birds, schools of fish and herds of land animals. The algorithm takes into account groups of independent agents which have very simple behaviour, mostly consisting on following its neighbors. This means agents have only a very limited and local view of the whole system. Even when the rules are simple, the ensemble motion of these agents can create complex dynamics.
Now let's assume that we have a function with several parameters and multiple values for these variables. The set of all the combinations of values that these parameters can have is

called the parameter space. In order to use swarm intelligence to find the optimal value of this function (global maxima or minima), we allow a group of independent agents to explore possible parameters in a space and move towards the optimal value by learning from experimenting different regions of the space. In this algorithm each agent is initialized in a random position in the parameter space. The algorithm progresses iteratively in epochs like many other machine learning algorithms. On each epoch, each agent calculates a velocity using its experience and the experience from its neighbors. The agent then moves towards the best visited position so far by it or its neighbors with the calculated velocity. As a difference to other algorithms such as genetic algorithms or differential evolution, this algorithm has memory in order to know the best visited position so far [88].
The general form of this algorithm [154] can be described as:

$$v_{k+1} = \vec{a} \otimes \vec{v}_k + \vec{b1} \otimes \vec{r1} \otimes (\vec{p1} - \vec{x}_k) + \vec{b2} \otimes \vec{r2} \otimes (\vec{p2} - \triangleleft \vec{x}_k), \tag{3.1}$$

$$x_{k+1} = \vec{c} \otimes \vec{x}_k + \vec{d} \otimes \vec{v}_{k+1} \tag{3.2}$$

Where the symbol $\otimes$ is the cross product, $\vec{v}_k$ is the velocity vector at time k, $\vec{a}$ is a momentum factor, $\vec{p1}$ is the particle's previous best position and $\vec{p2}$ is the globally best position in the whole swarm. Coefficients $\vec{b1}$ and $\vec{b2}$ define the strength of the attraction between particles. x_k is the position of the particle at time k and is updated based on the velocity using the coefficients $\vec{c} and \vec{d}$. $\vec{r1}$ and $\vec{r2}$ are random numbers.
This algorithm has been improved through time. Some examples of variations of the original approach improved for multi-objective optimization, efficiency, deep learning and data driven algorithms can be consulted in Ali and Kaelo [6], Blackwell and Branke [13], Eberhart et al. [44], Taormina and Chau [143].

Ant colony optimization Ant colony is an algorithm based on the movement of ants and how they find minimum paths to bring food into their colonies. This algorithm was first suggested by Marco Doring in [39]. The algorithm uses a fading trace inspired in the pheromones that ants segregate in order to efficiently find the shortest path in a graph. The concentration of the pheromones on each potential path from the origin (node A) to the hive (node B) is sensed by each ant. The pheromone trail $\tau_i j$ between node i and j is updated every time step with an exponential decay $\tau_{ij} = (1-\rho)\tau$ where $\rho in (0,1]$ and increased by a constant every time an ant visits the node

j coming from node i $\tau_{ij} = \tau_{ij} + \delta\tau$ [40].
An ant k located in node i will decide which node from the set N_i of nodes connected to i to visit next based on the following rule:

$$p_{ij}^{k} = \begin{cases} \tau_{ij} & j \in N_i \\ 0 & j \notin N_i \end{cases} \tag{3.3}$$

Longer travel times on each path contribute to a lower density of pheromones as they evaporate with time. Higher concentrations mean that more often ants have passed thorough it. This eventually leads to finding shortest paths between food and the hive. This can be translated to finding optimal connections between two nodes in a graph.

Evolution strategies There are several algorithms which fall in the category of evolution strategies. All of them are black box heuristic search algorithms inspired by evolution.
Genetic algorithms (GA) are one of the most popular gradient free optimization algorithms used today. Designed by John Holland and based on the evolutionary process or natural selection [73], GA starts with a random population of individuals (sometimes called chromosomes) which encode unique instances of the problem with specific parameters. These individuals are assessed for their fitness, using a measure of their optimality to fulfill the designated task. Once this fitness is calculated, it is used to identify the best individuals in the population. Depending on the implementation, a portion of the best individuals is passed to the next generation. In some proportion, individuals are also combined and mutated to form the new generation. The new generation is expected to preserve the best features of the best individuals while still allowing exploration of the parameter space.
The pseudocode below illustrates a basic implementation of GA.
Other forms of evolution strategies, like the one used in [130], are based on a progressive update of a parametrized search distribution over the parameter space. This means that the populations here are represented as probability distributions instead of groups of specific instances of parameters like in GA. By evaluating the fitness of a population by sampling from the current search distribution, parameters are updated to increase the overall fitness of the subsequent population, following a gradient in parameter space. Evolution strategies have recently been shown to work effectively in high-dimensional parameters spaces and to be well parallelizable.

```
Initialize the first generation with a population of N individuals, each with random values
  for the M parameters to be optimized;
while stop criteria is not met do
    Measure the fitness of the individuals in the current population;
    Select individuals with best fitness and take them to the new generation;
    Select a set of random individuals from the current generation and take them to the
      new generation;
    Create a new set of individuals by combining parameters from the best individuals;
    Mutate with low probability the individuals in the new generation;
end
```

Gradient descent Gradient descent (GD) is an optimization algorithm which is based on the calculation of the gradient of the function to be optimized starting from a random point in the parameter space [26]. The largest gradient, meaning the direction of the largest improvement of the value of the function, is then taken as next step to explore the surrounding parameter space. This is iteratively performed until no improvement is found, meaning either a global or local mimima is found, or a set of steps has been achieved. Gradient descent needs that the function to be optimized is differentiable and that the right step size (also known as learning rate) α is chosen such that the exploration is successful. The partial derivative of the fitness function with respect to the different parameters p_i to be optimized is calculated as follows:

$$\frac{\partial}{\partial p_i} F(p_i) = \nabla_{p_i} F, \tag{3.4}$$

and on each iteration the value of each parameters is updated using the following equation:

$$p_i = p_i - \alpha \nabla_{p_i} F \tag{3.5}$$

Currently, gradient optimization algorithms are the most widely used methods in machine learning. A vast family of algorithms has been developed from this basic concept, aiming at improving their performance, reliability, robustness and adaptation to ill-posed problems. Back propagation (BP) is a derivation of gradient descent which was first developed for continuous control [86] and then ported to artificial neural networks [160]. Today, different implementations and forms of BP are the preferred method for training or optimizing the weights of deep neural networks because BP can lead to high and robust performance, and training is very efficient compared to other methods. Some popular ML frameworks like Tensorflow [1, 2] and Pytorch [118, 117] provide automatic differentiation already integrated into their network models, which makes training easy and efficient. It consists on propagating error gradients layer by layer into the network in order to change the connections according to their contribution to the generated output. Because of its nature, BP is very well suited for feed forward neural networks with activation functions without discontinuities. An interesting extension of the BP algorithm is the so called Back Propagation Through Time (BPTT) [159], which is able of working with recurrent neural networks by unrolling them in time and creating equivalent feed forward networks.

Simulated annealing Inspired by statistical mechanics, in 1983 Kirkpatrick et al. [90] introduced the concept of optimization by simulated annealing. Annealing is a process in metallurgy in which metals are heated to a high temperature and then cooled gradually. This process makes metals more resistant and lowers the amount of defects because the slow temperature reduction allows atoms in the metal have time to find their most stable configuration. The optimization algorithm is based on the concept that finding the low temperature state of a system using a measure to calculate its energy, is an optimization problem which can be ported by analogy to other models. This algorithm is useful for multivariate optimization as it changes the different parameters in the system with a changing speed depending on the stability of the model.
The pseudocode illustrating a basic implementation of simulated annealing follows: In this

```
Initialize the system at a random point in the parameter space $p = p_0$;
while stop criteria is not met do
    Select a random point in the parameter space $p_{new}$ located in the neighborhood of the
      current location $p$;
    if $P$ then
        (
    end
    E(p), E($p_{new}$), T) ¡ R(0,1): $p = p_{new}$ Update the temperature T according to the
      annealing schedule;
end
```

pseudocode $P(E(p),E(p_{new}),T)$ is an acceptance probability function which depends on the ”energy” of the system (measure to be optimized) at the current and new selected point in the parameter space and on the temperature T. R(0,1) is a random value uniformly drawn between 0 and 1. The annealing schedule is the selected method to update the temperature as the optimization process evolves. Examples of such a schedule are a simple linear reduction or an update depending on a percentage of the total parameter space explored.

Multi-objective optimization Some optimization problems require that the system satisfies not only one but multiple targets. These targets are defined by measures of different features of the output produced by the system. Some of these targets might depend on each other or have competing effects, meaning that reducing the error on one will increase the error on one or several others. Finding the pareto front of a system is the ideal solution for a multi-objective optimization. The pareto defines the border where reducing the error on any single measure will as a consequence make one or more measures worst. A difficult part of multi-objective optimization is defining the right collective measure which considers all the objectives and really reflects their priority in the optimization process. A general weighted fitness function for multi-objective optimization can be described as follows:

$$F(m) = \sum_{i=1}^{N} w_i m_i, \sum_{i=1}^{N} w_i = 1 \tag{3.6}$$

where m is set of N target measures considered in this function and w_i is a factor which indicates the importance of the measure m_i for the overall fitness of the system. A fitness function could consider all objectives equally important or highlight some of them more than others. The way this measure is formed will directly impact the performance of the optimization process and the final result.

Learning as an optimization problem The definition of learning according to Mitchell [109] is:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
By experience here it is meant the execution of the task and the feedback that is given based on this execution and its results. Algorithms, such as neural networks, are trained to perform a task by optimizing their parameters or internal configuration in order to minimize the error between the produced and the desired output. Several optimization algorithms have been developed, tuned, enhanced and extended to fulfill requirements from machine learning. This is because the success of algorithmic learning heavily relies on making an update in the system which improves the results of future executions of the task and interactions with the environment. In other words,

in order to improve the result the learning system should aim at optimizing its performance and, as a consequence, reducing the error between the actual and the desired result. In fact, nature tends to optimize resources whenever possible. This is even reflected in evolution, where optimal adaptation to the environment is rewarded with survival and progeny. Optimization is key to learning because, by evaluating the result of several actions, one should learn which action produces “more optimal“ results.

3.1.1 Hyperparameter optimization

Each optimization algorithm has parameters on its own. These parameters help adapt the algorithm to specific applications and in most cases it is not intuitive how their initial conditions must be set. They impact the overall efficacy of the algorithm and, as a consequence, most of the time need to be manually tunned by an expert. Hyperparameter optimization is the term used to define the optimization of the parameters of the optimization algorithm. It is process which considers two nested optimizers, where the goal of the first optimizer remains to minimize the error of the system, while the goal of the second is to optimize the performance of the first optimizer.
Learning to learn [150] or meta learning is a hyperparameter optimization paradigm which aims at optimizing learning performance. It consists of generating training protocols which interleave similar tasks and perform learning updates at different scales. It leaves slow, usually structural, changes to an external optimizer which favors generalization and relies on fast updates from the inner optimizer or learning system to adapt and learn specific tasks. It has been proposed as a way to achieve few shot learning of new tasks and can be combined with methods to avoid catastrophic forgetting [8, 72, 156]. Learning to learn can be seen as a paradigm to dynamically improve learning performance by exposing the system to a family of tasks which share common traces. This method relies on the ability of the system to transfer previously learned and useful configurations or parameters from one task to another.

3.2 Structural plasticity, optimization and control

Nature has taken millions of years to develop and optimize our brains to perform specific tasks. Our brain is the results of years of reconfigurations, tests, successes and failures. Today it is still unclear to us how the mechanisms for adaptation, learning and brain recovery among many other complex functions of the brain really work. The ability to adapt and learn or relearn a capacity

damaged through a lesion is believed to be linked to changes in the networks that form the brain. How these changes have come to be defined and constrained is a question of evolutionary optimization. As it has been said before, neural networks change their structure as they are generated, as a response to external stimuli and as a response to other structural and biochemical changes. One can imagine that the set of rules which guide structural and synaptic plasticity has a defined goal, which is reflected in the robustness, flexibility and capacity of our brains.
The Webster dictionary defines *optimization* as the ”an act, process, or methodology of making something (such as a design, system, or decision) as fully perfect, functional, or effective as possible; specifically : the mathematical procedures (such as finding the maximum of a function) involved in this.” Optimization in mathematical terms refers to finding a minima or maxima of a function. Control theory deals with the steering of continuously operating dynamical systems. It is a subfield of mathematics which describes systems using a transfer function which describe how input is transformed into output. Changes can then be made to such systems in order to modify their output. The goal is to optimize the performance of a system by reducing the error between its real and a desired output. It is considered here, that the output of the system can be observed and that this value changes when the system’s parameters and input are modified. The error is then a function of the inputs and the system’s internal configuration. The shape of this function and its characteristics impact greatly the ability of a control algorithm to find the optimal solutions for a given system. This function can be highly degenerate, meaning that there are several parameters in the system which lead to the same value for the chosen measure. The function can also have several local and global minima or maxima which leads to finding only sub-optimal solutions.
While control theory deals with the problem of taking a dynamic system progressively towards a desired output, optimization deals with finding the best configuration of a system which provides the desired output under the measure of the selected fitness function. In this thesis I use the concept of a controller as an element external to the observed system which can perform a dynamic optimization process on it. My motivation to see plasticity from the optimization perspective is that this phenomenon aims at taking a neural network from an initial state A to a final state B, where the error between the actual and desired output of processing a given input is ”hopefully” minimized. I treat the brain as a system with a dynamic structure, which can be modified and in which several mechanisms of control are simultaneously working to obtain an specific output. In this thesis the focus is on structural plasticity but its combination with synaptic and other means of plasticity is essential to better understand and study the brain as an organ

capable of learning, adapting and healing. A working theory is that the essential features of how this optimization process takes place, seem to have been refined by evolution and imprinted into the plasticity rules we try to elucidate today.

3.2.1 General form of network dynamics

In [114], we introduced a formalization of neural networks which can be used to understand them as a controllable and observable system. Let a neural network be defined by a set of ordinary differential equations in which $x_1(t), x_2(t) \ldots x_n(t)$ are state variables of the system at time t. It is assumed that neurons in this model can be either in an active or quiescent state. The master equation of a neural network has been derived and explained in Cowan [29] and Ohira and Cowan [115]. This equation provides a mathematical description of the evolution of stochastic neural networks in the form of a Liouvillian:

$$L = \alpha \sum_{i=1}^{N} (\Delta_{+i} - 1)\Delta_{-i} + \sum_{i=1}^{N} (\Delta_{-i} - 1)\Delta_{+i}\,\phi\left(\frac{1}{n_i}\sum_{j=1}^{N} \omega_{ij} x_j\right) \tag{3.7}$$

where α is the decay function after a neuron has spiked, Δ_{+i} and Δ_{-i} are the raising and lowering operators which take a neuron i to and from an activation state, n_i is the number of connections to neuron i, N is the total number of neurons in the network, ϕ is the activation rate function which depends on the neuron model and ω_{ij} is the strength of the connection between neuron i and j. Synaptic growth and connectivity variations in neural networks further increase the complexity of the system. In the case of variable connectivity, the network master equation is transformed into:

$$L = \alpha \sum_{i=1}^{N} (\Delta_{+i} - 1)\Delta_{-i} + \sum_{i=1}^{N} (\Delta_{-i} - 1)\Delta_{+i}\,\phi\left(\frac{1}{n_i\big(u(t)\big)}\sum_{j=1}^{N} \omega_{ij}\big(u(t)\big)\, x_j\right) \tag{3.8}$$

where both ω_{ij} and n_i depend on the control signal u coming from the synaptic and structural plasticity algorithms at time t. This formulation is introduced to expose variables $u(t)$ in the system, which can be controlled. Modifications in these signals induce changes in the network and provide means to achieve a target dynamic profile. However, it is worth noting that this approach is also applicable to non-stochastic neural networks.

3.2.2 Control for network state trajectories

Both synaptic and structural plasticity can be seen as biological multivariate controllers. Under this view, the system gradually creates and destroys connections between neurons, or modifies the strength of existing synapses (control), to achieve a transition from one initial state to a final steady (or even homeostatic) state. This final state can be a previously known activity state which has been altered, as in repair after a lesion, or a new activity state to be achieved, as is the case in learning. Thus, the evolving connectivity problem can be mathematically expressed in terms of control theory as defined in Kirk [89].

In this case, the control signals refer to the variations in the connectivity of the network while the states refer to the dynamics of the network. The state equations take the form of:

$$\dot{\mathbf{x}} = \mathbf{a}\big(\mathbf{x}(t), \mathbf{u}(t), t\big) \tag{3.9}$$

where $\mathbf{u}$ is the history of control signals during the interval $[t_0, t_f]$, and the state trajectory denoted by $\mathbf{x}$ is the history of state values during the same time interval. A control history which satisfies the constraints of the system (in this case, experimental parameters of neurons and synapses) during the time interval of interest is called an ‘admissible control’. On the other hand, an ‘admissible trajectory’ is a state trajectory which satisfies the constraints of the state variables through the whole period of interest. The final state of the system is then required to lie in a specific region, defined as the target set, of the $n+1$-dimensional state-time space.

Figure 3.1 shows the diagram of a single neuron from the control perspective. The activity of this excitatory neuron is determined by a nonlinear transfer function $f_E(V)$ acting on the membrane potential V. The excitatory neuron receives positive input in the form of a Poisson spike train $P(t)$ filtered by $h_e(t)$. It also receives positive input from the excitatory population and negative input from the inhibitory population which is weighted by the number of incoming connections c_1 and c_2 respectively and filtered by $hi(t)$ and $he(t)$. $he(t)$ and $hi(t)$ therefore denote the excitatory and inhibitory post-synaptic potentials, respectively. The excitatory activity is weighted by $c1$, filtered by $he(t)$ and fed into the inhibitory and excitatory populations. The controller (e.g. structural plasticity algorithm) $C(\varepsilon - \lambda)$ translates the error between the current firing rate of the neuron λ and the desired firing rate ε into a control signal $u(t)$. This signal changes $c1$ and $c2$ accordingly in order to modify the incoming connections to the neuron.

Using a matching terminology, Figure 3.2 shows the diagram of a two population neural network

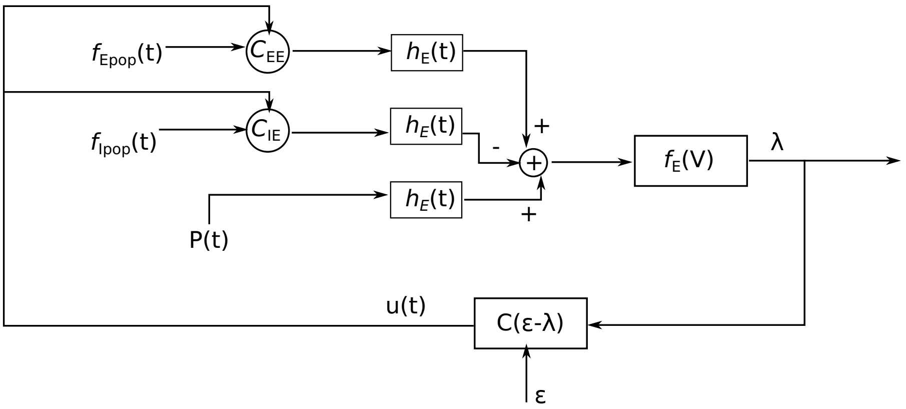


Figure 3.1: Diagram of a excitatory neuron from the perspective of control engineering. Based on [16]

from the control perspective.

There has been previous work which analyzes homeostatic changes in neural networks form the perspective of control theory. In [61], the authors explore the concept of homeostatic control in order to study stability of recurrent neural networks. They show an analytical framework to study the ranges of parameters which allow stable integration of control signals following homeostatic procedures in the presence of different degrees of recurrence.

In [16], the author analyses the impact of changes in the connectivity with a sensitivity analysis against the firing rate and power spectrum of the networks. In order to do this, she used mean field theory to describe the ensemble behaviour of neural populations and reduce the degrees of freedom from inter neuron connections to inter population connectivity. While Bos et al. [16] made systematic manual variations of the connections in the networks, in this thesis I use structural plasticity to explore different connectivity setups. Nevertheless, this work sets an analytical basis for the numerical simulations shown in this thesis.

By applying the control signal $\mathbf{u}(t)$ from t_0 to t_f, the system will evolve from its initial state x_0 following some trajectory to a final state x_f. The 'performance' of this trajectory is the difference between a desired and the obtained measure for a heuristic involving the dynamics of the system. In this case, the performance function is given by the homeostatic rules the system must follow. To reach a defined target activity regime, we cannot know a priori whether an optimal admissible control exists, which leads the system through an admissible trajectory for a given performance

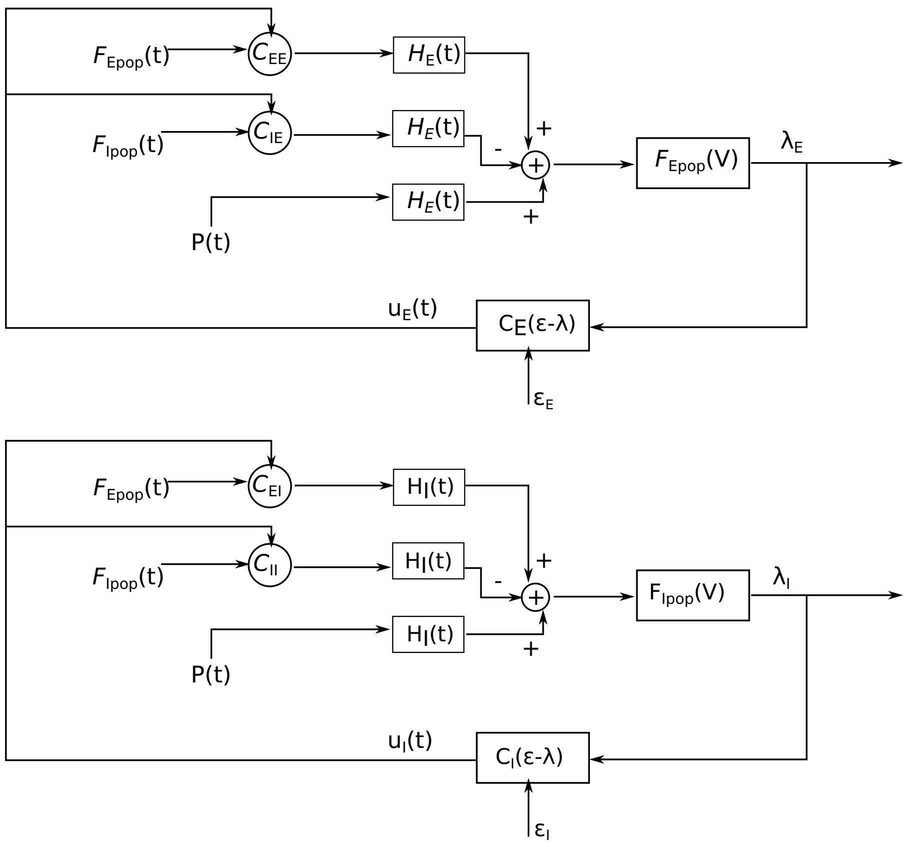


Figure 3.2: Diagram of two populations from the perspective of control engineering. Based on [16]

function. It may be impossible to find such a control history, and even if it exists, it may not be unique or numerically stable. Structural plasticity seeks for one or more admissible trajectories of the system. For the class of neural networks described by the dynamical equations above, the problem of finding the exact control signals or free parameters for a simulation leading to experimental results cannot be solved in polynomial time. However, it may still be possible to confirm solutions in polynomial time.

3.2.3 Structural plasticity as an optimization algorithm

Even though the structural plasticity algorithm is particularly suitable to optimize the connectivity of a neural network to fit given dynamical properties, it could be applied to problems in other domains. Structural plasticity is similar to simulated annealing in the sense that the strength of the changes (rate or step size) in the parameter space changes progressively as the system approaches the target. Usually, the growth rate of the synaptic elements in the homeostatic rule lowers as the solution is closer, just as temperature lowers in the simulated annealing algorithm. It works as swarms because each neuron in the network, while having individual rules, indirectly follows and is followed by all neighboring neurons (those with synaptic contacts to it) with a given strength (synaptic weight). Its random nature also points towards Bayesian sampling as it has been previously highlighted in Kappel et al. [81] and Bellec et al. [11]. Structural plasticity involves the whole process of moving the neural network from an initial state to a final state with new dynamical features which reduces an error in the desired measure of an output. Because of this, structural plasticity works as a homeostatic controller of the connectivity in the brain.
In their work, Kappel et al. [80] propose structural plasticity as a Bayesian inference algorithm which, combined with STDP, allows the integration of priors into the network development through time. Their focus on structural plasticity as machine learning algorithm is on the level of network learning and function. In this thesis, my focus is on defining structural plasticity as an algorithm for the generation of connectivity taking into account local optimization by each neuron and ensemble learning achieved by collective optimization. An interesting feature of structural plasticity as optimization algorithm is that it can be used to solve multi-objective optimization techniques. Neurons individually seek to reach their goal by reducing the error between their current activity and a desired activity. The synapses created between neurons define dependencies with other objectives. If combined with synaptic plasticity, this algorithm could provide a powerful tool to model different kinds of multi-objective optimization problems which do not need to be directly linked to neuroscience or brain simulation.

3.3 Summary

The goal of this chapter was to provide the reader with a broad view of the available optimization algorithms in the computer science community and link some of their characteristics to the structural plasticity algorithm. This will be used in Chapter 7 to explore the potential of structural plasticity to be used as an optimization algorithm for different problems involving changes in the input and output of dynamic processing units. I also presented structural plasticity from the control theory perspective. This is a useful change from the biology perspective to the engineering perspective in order to aid a systematic study of the sensitivity of the system, in this case a neural network, to perturbations in its structure. With these concepts we will move to the next chapter to the implementation of software tools which are designed to aid a better understanding of simulations of spiking networks with a dynamic connectivity.

4 Implementation and usage of a structural plasticity algorithm in NEST

The previous two chapters provided a general overview of terms, models, and algorithms in the fields of neuroscience, computational neuroscience, and computer science which are useful to understand my work on structural plasticity. In this chapter, I present how the structural plasticity model proposed by Butz and van Ooyen [22] was implemented in the neuronal network simulator NEST [58] in order to create self organizing large scale neural networks, as described in Diaz-Pier et al. [37]. The scalability of the implementation is evaluated and the performance of the model on two use cases is analyzed. The implementation presented here is capable of self-organizing the connectivity within a cortical microcircuit model consisting of $100,000$ neurons in total, starting with a fully disconnected setup. The scenario where partial information of the connectivity is given as initial condition and an stable connectivity pattern is obtained in the end is also presented. This chapter also discusses the changes performed to the code since its first release and into the 5g simulation kernel of NEST. The work I present here constitutes the first step required to simulate structural plasticity and explore its applications.

4.1 Implementation of a structural plasticity model in NEST 2.10

The implementation of the structural plasticity algorithm described in this chapter is based on the version 2.8 of the NEST software [49] and first included in release 2.10 [17], creating a novel possibility for automatic generation of connectivity in large-scale neuronal networks. Please notice that in the following the mean firing rate is used instead of calcium concentration to measure the electrical activity of the neurons as used in Diaz-Pier et al. [37] and the original model by Butz and van Ooyen [24].
In accordance with the original formalization described in section 2.1, the algorithm consists of

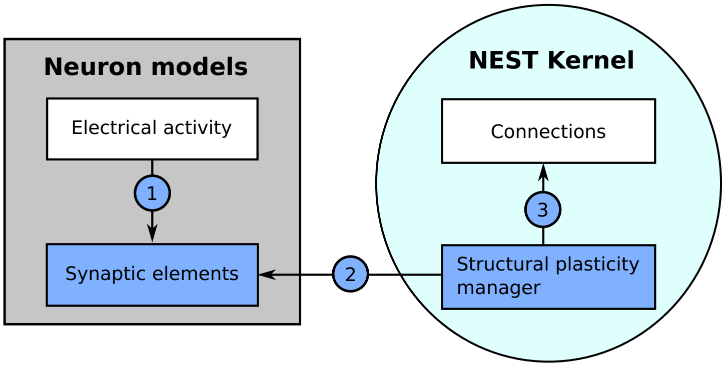


Figure 4.1: Diagram of the implementation of the structural plasticity model in NEST. In 1) the number of synaptic elements is calculated depending on the electrical activity of the neuron. These calculations are optimized using MPI and OpenMP. In 2) the structural plasticity manager gathers the number of synaptic elements per neuron using MPI directives and in 3) creates or deletes synapses to update the connections between neurons using MPI and OpenMP.

three repeating parts which can be visualized in a general form in Figure 4.1 and described as follows:

1. Update in electrical activity. The firing rate λ at time t is calculated by low-pass filtering spike train data by convolving that data with an exponential decay kernel [116]: the current firing rate λ is increased by a constant $1/\tau$ for each spike and decays exponentially with a time constant τ between firing times. Thus,

$$\tau \frac{\mathrm{d}\lambda}{\mathrm{d}t} = -\lambda + \sum_{t^f} \delta\left(t - t^f\right) \tag{4.1}$$

   where t^f are the firing times of the neuron and δ is the Dirac delta function. The constant τ can be set by the user but the default value is $\tau = 10$ s. The *Archiving_Node* class, which is the general interface for all neurons, was modified and new variables to store the of the firing rate and the τ were added. The *Archiving_Node::set_spiketime* method was also modified to update the firing rate according to the first case defined in Equation (2.1), which is performed at every time the neuron spikes. The Node class was modified by adding the method *Archiving_Node::update_synaptic_element*. This method updates the firing rate according to the second case defined in Equation (2.1). This method is called by the *Scheduler* class when every synaptic update interval is reached.

2. Update in synaptic elements. The first step taken in order to design and develop a framework for synaptic elements (e.g. axonal boutons and dendritic spines) was to redefine synapses in such a way that they can now be described using connection elements. This description can be applied to every available neuron model in NEST for generating electrical activity. The design also considers that the users can define their own synaptic elements and their corresponding growth dynamics. The class *SynapticElement* was created in order to represent the connection points for the neurons. The class *GrowthCurve* was also created in order to define the homeostatic rules which guide the creation and deletion of synaptic elements. Currently, the available growth dynamics are based on either a linear or a Gaussian growth curve. The linear growth curve uses an exact integration method to update the number of synaptic elements, while the Gaussian growth curve uses a forward Euler integration method. The framework can be further extended by the user to incorporate more complex element growth dynamics models. An example of such curves is shown in Figure 4.5, where independent dynamics for each type of element in a network of 8 populations (see use case on cortical microcircuit in Section (4.1.2)) have been defined. Synaptic elements are used as a discrete value, the actual number of available synaptic elements is an integer truncated from the float variable used to represent them. The *Archiving_Node* class now incorporates a map data structure to store the synaptic elements. The method *Archiving_Node::update_synaptic_element* takes care of updating the number of each of the synaptic elements in the map using the value of the firing rate at the time of the call and the corresponding growth curve. The method *Archiving_-Node::decay_synaptic_element_vacant* takes care of deleting a percentage of the unused or vacant synaptic elements on every call. Both methods are called by the *Scheduler* at the end of every synapse update interval.

3. Update in connectivity. To coordinate the changes in the structure of the network, a new *StructuralPlasticityManager* class was implemented. At the end of every synapse update interval, the *Scheduler* calls the the newly implemented structural plasticity connectivity manager via the *Network* class. The *StructuralPlasticityManager* determines, for each neuron, how many vacant synaptic elements are available for new synapse formation and how many deleted synaptic elements caused synapse breaking. Then it makes use of the *ConnBuilder* in order to create or delete connections. For this, the *ConnBuilder* was extended to include the new methods *ConnBuilder::sp_disconnect_* and *ConnBuilder::sp_-*

*connect_*. Once new synapses are formed, synaptic elements are tagged from "vacant" to "connected". It is important to notice that when a synapse breaks due to the deletion of one synaptic element, the counterpart remains and becomes vacant again. This remaining counterpart can form a new synapse at the next update in connectivity. This effect preserves the network rewiring capabilities of the original formulation. A detailed diagram of how the new calls are integrated into the normal simulation flow of NEST can be seen in Figure 4.2.

An important feature implemented to simulate structural plasticity in NEST is the ability to create and delete synapses during the simulation time. The implementation of the connection management overcomes the limitation of the NEST simulator that currently models networks with a fixed connectivity. The dynamic creation and deletion of synapses was implemented using the connection framework released in version 2.6.0. This connection framework improved memory usage to store connection data and reduces the computation time needed to create a connection. The main limitation of the structural plasticity algorithm described by Butz and van Ooyen [22] is that it requires global knowledge of the synaptic elements of the entire network. Fortunately, the MPI global communications, also used by the NEST kernel to communicate the electrical activity between the neurons during the simulation, do not pose a substantial bottleneck since changes in connectivity are assumed to take place on average around a factor of 100 times slower than changes in electrical activity. Therefore selecting a biologically realistic growth rate of around 10^{-4} elements/ms will result in an exchange of data that is sufficiently low rate so as not to impact the scalability of the simulator as a whole. At the end of each connectivity update step, the number of created/deleted synaptic elements per neuron are communicated to all MPI processes and a global shuffle subsequently assigns the new pairs of neurons that should be connected, and likewise chooses existing connections for deletion.
The original model from Butz and van Ooyen [22] considers distance dependency when choosing the targets to create new synapses. In this implementation, target neurons for new synapses are chosen in a random fashion, independently of their distance or any other factors. The probability of two neurons connecting to each other depends solely on the number of available compatible synaptic elements between them. When deletion occurs, a random synapse from the neuron is chosen. There are no further considerations about the nature of the synapse or its strength.
The actual creation and deletion of the synapses is finally done in parallel using the NEST connection framework. As stated before, a single update in connectivity should not produce a

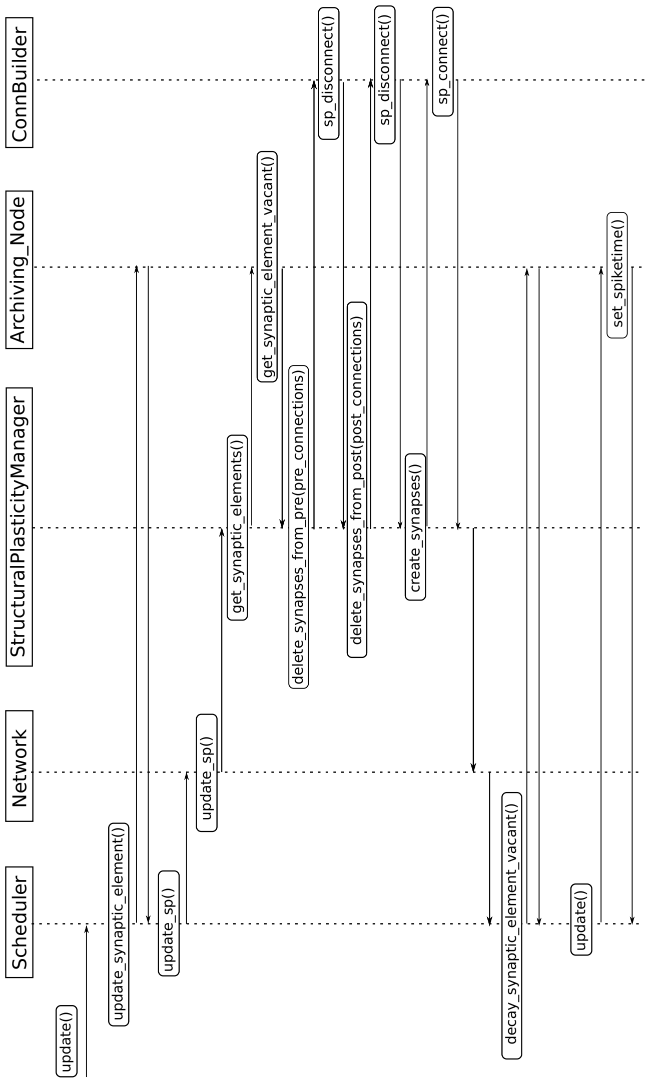

Scheduler
Network
StructuralPlasticityManager
Archiving_Node
ConnBuilder
update()
update_synaptic_element()
update_sp()
update_sp()
get_synaptic_elements()
get_synaptic_element_vacant()
delete_synapses_from_pre(pre_connections)
sp_disconnect()
delete_synapses_from_post(post_connections)
sp_disconnect()
create_synapses()
sp_connect()
decay_synaptic_element_vacant()
update()
set_spiketime()

Figure 4.2: Integration of the new structural plasticity calls into the normal simulation flow of NEST

major modification of the network. That means that only a small part of the neurons should create or delete a synaptic element between two updates in connectivity.
It is important to highlight that the usage of global communication is a characteristic of the technical implementation of the algorithm and is not related to the functionality of the model. If topology was to be taken into account, the ability of a neuron to connect to any other would be limited by the constraints imposed by its relative position to others. Global communication would still be used by the implementation, but only relevant information would be taken into account to define the connectivity. The local homeostatic rules only define the creation or deletion of synaptic elements per neuron. The number of available synaptic elements is transmitted globally and the synaptic plasticity manager takes care of forming new synapses or deleting existing ones based on this information.
The update of electrical activity and of the number of synaptic elements is performed by every individual neuron and therefore benefits from the parallel framework already implemented in NEST. Indeed, the NEST software has already demonstrated its high scaling properties on supercomputer, including the JUQUEEN system [64, 93].
Finally, the Python interface of NEST (PyNEST [48]) was extended to allow users to easily set up the structural plasticity parameters. It is important to highlight that the user can enable structural plasticity inside the simulation and then disable it when the network has achieved a desired connectivity pattern or activity level. The user can now also delete synapses even without enabling structural plasticity, in a similar way as the connect functions work in NEST.
While supercomputers are required for very large-scale simulation, smaller networks can also be run on a personal workstation or laptop according to the NEST development philosophy. This is a fundamental advantage of this implementation of structural plasticity in terms of capacity to test different configurations, as it provides high flexibility and portability for the neuroscientist. Later, changes in the implementation were performed as the NEST kernel code evolved through its 2.16 release. Finally, with the arrival of the 5th generation kernel in NEST, substantial changes were implemented in order to continue to support the functionality.

Setting up a network in NEST with structural plasticity

This section introduces the high level functions that are have been added to NEST with the structural plasticity framework using PyNEST.
In order to set up the network using structural plasticity, one first needs to define the time at which

updates in the structure of the network should take place as follows:

```
nest.SetStructuralPlasticityStatus({
  'structural_plasticity_update_interval':
    update_interval,
})
```

The next step is to define the synapses which can be dynamically modified by the structural plasticity manager during the simulation. This is achieved by:

```
nest.SetStructuralPlasticityStatus({
  'Structural_plasticity_synapses':{
    'structural_plasticity_synapse_ex':{
    'model':'structural_plasticity_synapse_ex',
    'post_synaptic_element':'Den_ex',
    'pre_synaptic_element':'Axon_ex',
    },
    'structural_plasticity_synapse_in':{
    'model':'structural_plasticity_synapse_in',
    'post_synaptic_element':'Den_in',
    'pre_synaptic_element':'Axon_in',
    },
  }
})
```

Here, two types of synapses are being defined, one for the excitatory synapses and another one for the inhibitory synapses. It is important to notice that in this definition, a name for the post and pre- synaptic elements is also specified. This allows the structural plasticity manager to create new synapses of the type specified in model when synaptic elements related to this label become available. This way of setting up the dynamic synapses also allows the user to define static connectivity constraints in the network. This can be achieved by using one synapse model which is not registered for structural plasticity to define this fixed connectivity. For the moment, no other constraints in connectivity like indegree or outdegree ranges can be specified. Nevertheless, thanks to its flexible design, the model can be extended to add new constraints.
The next step involves the specification of the growth curves for the synaptic elements defined above. This is done as follows:

```
growth_curve_e_e={
  'growth_curve':gaussian,
  'growth_rate':0.0001,
  'continuous':False,
  'eta':0.0,
  'eps':5.0,
}
```

This is an example of a Gaussian growth curve where the minimum firing rate required to start generating synaptic elements is $\eta = 0.0\,\mathrm{Hz}$, and the desired firing rate is set to $\varepsilon = 5\,\mathrm{Hz}$. Finally, the rate at which the synaptic elements will grow is $\nu = 1 \times 10^{-4}$ elements/ms. Independent growth curves can be created for each synaptic element.
Now that the growth curve has been defined, it can be assigned to the synaptic elements that each neuron will be able to grow. After that, the neurons can be created and NEST must be notified that these synaptic elements are linked to the neurons:

```
synaptic_elements = {
  'Den_ex':growth_curve_e_e,
  'Den_in':growth_curve_e_i,
  'Axon_ex':growth_curve_e_e,
}

nodes = nest.Create('iaf_neuron',number_excitatory_neurons)
nest.SetStatus(nodes,'synaptic_elements',synaptic_elements)
```

Here the creation of neurons pertaining to the excitatory population is shown. Each neuron has three types of synaptic elements, one dendritic excitatory, one dendritic inhibitory and one axonal excitatory.
The final step is to enable structural plasticity and simulate:

```
nest.EnableStructuralPlasticity()

nest.Simulate(t_sim)
```

A complete PyNEST example which describes how to create a network with two populations, enable structural plasticity and simulate the network is distributed together with the NEST simulator.

4.1.1 Scalability

Type of synaptic element	η	ε	ν
Excitatory neurons, excitatory elements	0.0	5.0	1.0×10^{-4} elements/ms
Excitatory neurons, inhibitory elements	0.0	5.0	1.0×10^{-4} elements/ms
Inhibitory neurons, excitatory elements	0.0	20.0	4.0×10^{-4} elements/ms
Inhibitory neurons, inhibitory elements	0.0	20.0	1.0×10^{-4} elements/ms

Table 4.1: Parameters for the homeostatic growth rules for each type of neuron and synaptic element in the two population network model.

To assess the scalability of the framework, I designed strong and weak scaling tests of the structural plasticity implementation. For all tests, networks with 80% excitatory and 20% inhibitory neurons were created. The parameters of the homeostatic rules for synaptic elements in the simulation were defined as shown in Table 4.1 on page 51. The decay constant was set to $\tau = 10000.0$ for all neurons. The post synaptic amplitude of individual synapses was set to 1.0 mV. External input was provided using a Poisson generator with a frequency of 10^4 Hz. The post synaptic amplitude of individual synaptic input was set to 0.01 mV. The simulation was run for 100 s, with a step size for the numerical integration of 0.1 ms. The updates in the network connectivity were performed every 10 ms. These values were chosen as they proved to be one parameter combination that allowed for stable self-organizing growth of the network towards the homeostatic equilibrium (See section 3.3.1 for additional comments on the selection of this parameter set).

Weak scaling tests were performed for networks with $5,000$ neurons per node and settings of 1, 2, 4, 8 and 16 nodes, each node using 28 cores. Strong scaling tests were performed with a network of $100,000$ on the same hardware configurations as the weak scaling tests. Only physical cores were used, no simultaneous multithreading was enabled. A hybrid optimization approach was chosen, in which MPI is used for communication between nodes and OpenMP for intra node communication. All measurements were performed on the JUROPATEST cluster, which provides up to 70 nodes (T-Platforms V210s Blades), each with 2 x Intel(R) Xeon(R) CPU $E5-2695$ v3 (Haswell) with 14-core processors (2.30 GHz) and 128 GB DDR memory, running with Scientific Linux release 6.5 (Carbon).

While the update in electrical activity has been proven to scale up to 10^9 neurons, it is important to verify that updating the number of elements and the deletion and formation of synapses does not restrict the expected scaling, at least in the desired regime of up to 10^6 neurons. Updates in synaptic elements and connectivity make use of MPI's ÁllGatherćommunication scheme to communicate the data. This collective communication is also used by the NEST kernel to communicate the spiking activity between the neurons during the simulation. Although AllGather implements communication between all processes, it is very unlikely that a huge amount of data has to be communicated when a reasonable growth rate of around 10^{-4} elements/ms because updating the number of synaptic elements and the connectivity are very slow processes compared to the update in electrical activity.

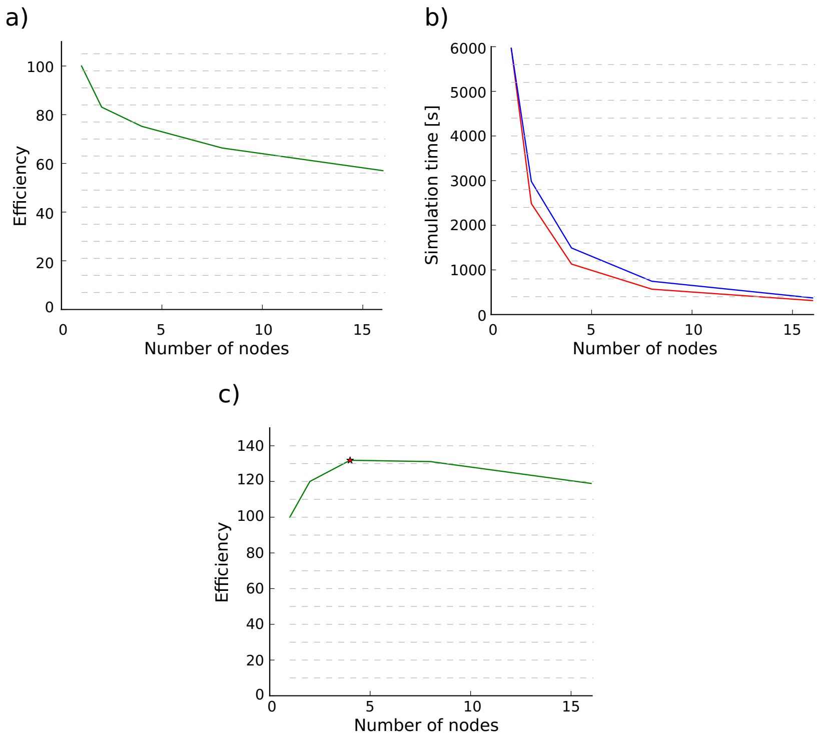


Figure 4.3: Results of the scalability tests performed with structural plasticity. A) Efficiency as a function of the number of nodes for 5,000 neurons in the weak scaling test. The network was allowed to grow synapses following the structural plasticity rules during a simulation of 100 s of biological time. B) Simulation time (red curve) as a function of the number of nodes for a network of 100,000 neurons and in the strong scaling test. The blue curve indicates ideal linear scaling. C) Efficiency as a function of the number of nodes for a network of 100,000 neurons in the strong scaling test. The peak scaling efficiency is marked with a star.

Weak scaling

Figure 4.3A shows the efficiency, defined as the speed-up divided by the number of nodes, of the implementation as measured by a weak scaling test with 28 OMP threads running on each node. It is visible that, as the number of neurons increases, so does the total number of synapses. The presence of new synapses leads to an increase of communication of structural changes between neurons, which leads to a decrease in the efficiency of the simulation.

Strong scaling for a network of 100,000 neurons

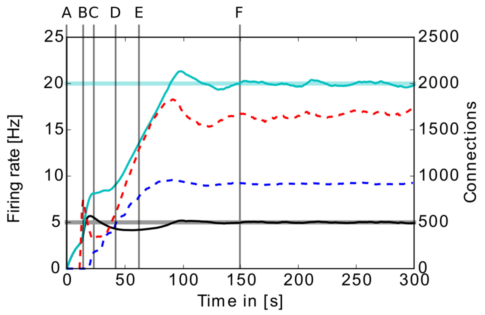
Figure 4.3B shows the computation times of the strong scaling tests for a network of $100,000$ neurons, and Figure 4.3C shows the efficiency of the strong scaling test. The peak efficiency is achieved with 4 nodes and 112 cores. These results show supra-linear scaling for this network. In Plesser et al. [120], supra linear scaling for biological neural networks on NEST was demonstrated due to increasingly efficient caching.
These results show that the introduction of the new structural plasticity framework into NEST has no impact in the scalability of the simulation up to a network size close to that of a cortical column if a suitably low growth rate is selected.

4.1.2 Performance on the use cases

The main objective of the structural plasticity framework is to provide the user with a tool to model the dynamic creation and deletion of synapses between neurons of a neural network in a scalable manner. There are several applications in which structural plasticity can be used. In this section I detail two use cases as examples. The first use case shows the basic functionality of the framework and how it can be used to study the relationship between connectivity and activity. I also show how this simple set-up can model critical development periods in the network connectivity. The second example is a more complicated case with several populations, where the objective is to show how connectivity can be self-generated in a network by using the synaptic element growth curves as connectivity fitness rules. All simulations were carried out with NEST version 2.8.0 extended by the structural plasticity implementation. As already mentioned, this implementation was then made public in the NEST release 2.10.

A simple two population network

In this initial use case, I generate a network with a total of 1000 leaky integrate and fire neurons, 80% excitatory and 20% inhibitory. Parameters for the growth rules of each type of neuron and synaptic element are the same as detailed in Table 4.1 on page 51. The connectivity in the system was allowed to evolve using a Gaussian growth curve for 3000 s, with an integration step of 0.1 ms and a delay of the connectivity update equal to 100 integration steps. The simulations were done on a workstation with 8 Intel core i7-4770@3.4 GHz CPUs with openSUSE 13.1.



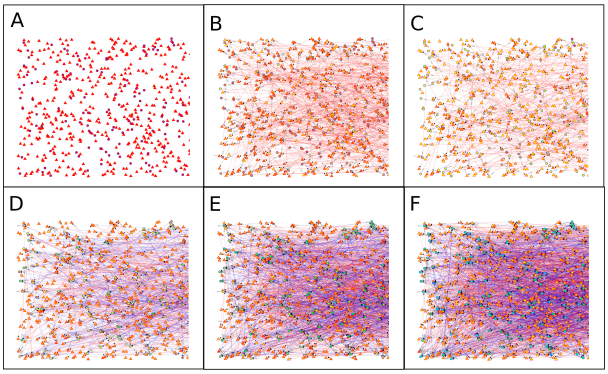


Figure 4.4: Upper panel: Firing rate and numbers of connections as functions of time in a simple two population network. The cyan and black curves show the firing rate measured in the inhibitory and excitatory populations, respectively. The paler horizontal lines indicate the corresponding target levels ε. The blue and red dashed curves indicate the total number of connections in the inhibitory and excitatory populations, respectively. Vertical gray lines indicate the times of the snapshots displayed in the lower panel. Lower panel (from A to F): Evolution of the connectivity in the two population network visualized using MSPViz (see Section 4.1.3). Images show half of the total amount of neurons in the network, where triangles represent excitatory neurons and circles inhibitory neurons. Red lines indicate excitatory connections while blue lines indicate inhibitory connections.

The upper panel of Figure 4.4 shows the evolution of the firing rate and total number of connections. The lower panel shows the evolution of the connectivity in the network using a visualization tool called MSPViz, specifically designed to visualize structural plasticity. For more on this tool see Section 4.1.3. During the first 30 s of the simulation, mostly excitatory connections are created (Figure 4.4A). This allows the firing rate to increase in both populations. When the target mean electrical activity is reached and overshoots in the excitatory population (Figure 4.4B), the number of excitatory connections starts to decrease (Figure 4.4C) until the desired firing rate is achieved and stabilized in the excitatory population. However, both pre- and post-synaptic elements in the inhibitory population are still being created because it has not yet reached its target mean electrical activity. It is important to remember that neurons have no information regarding the global status of the network and the evolution of their synaptic elements depends solely on the predefined homeostatic local rules. At around 40 s (Figure 4.4D), an increment in excitatory connections is triggered by the enhanced levels of inhibition. This leads to a complete rewiring of the network (Figure 4.4E). The trend is preserved until the mean electrical activity in the inhibitory population is also reached (Figure 4.4F).
In this network setting, the inhibitory population has a higher level of activity than the excitatory population. As previously mentioned, the probability of two neurons connecting depends only on the number of available compatible synaptic elements between them. At the start of the simulation, the inhibitory population must offer more post-synaptic elements for excitatory synapses than the excitatory population, otherwise the excitatory population would reach equilibrium first and cease to create excitatory pre-synaptic elements. As a result, not enough excitatory synapses would be created to the inhibitory population and it would never reach the desired level of activity. The structural plasticity parameter space is broad and a certain amount of exploration is required to discover combinations for the growth of each type of synaptic element which can take the network to equilibrium. However, there is in general no unique combination of parameters leading to equilibrium, and different equilibrium combinations will typically produce different connectivity patterns. At this point, biological constraints must be applied to choose between them.

The cortical microcircuit network

In this second use case, I create a four layer network based on the model of the cortical microcircuit proposed by Potjans and Diesmann [122]. Each layer contains one inhibitory and one excitatory population of leaky integrate and fire neurons. In the simulations presented here, the network starts with the same number of neurons in each population as in the previous study, but without any synaptic connections. For each population, I define a level of desired mean electrical activity based on experimental literature and a growth curve which defines the dynamics of the variation in the number of pre- and post-synaptic elements. These are Gaussian shaped curves with two intersections with the x-axis that determine the minimum amount of electrical activity required to form any synapse (η), and the target mean firing rate for the neuron (ε) as described in Chapter 2. The curves are illustrated in Figure 4.5.

In the first example, I tune the growth rate to achieve an stable growth regime for the network

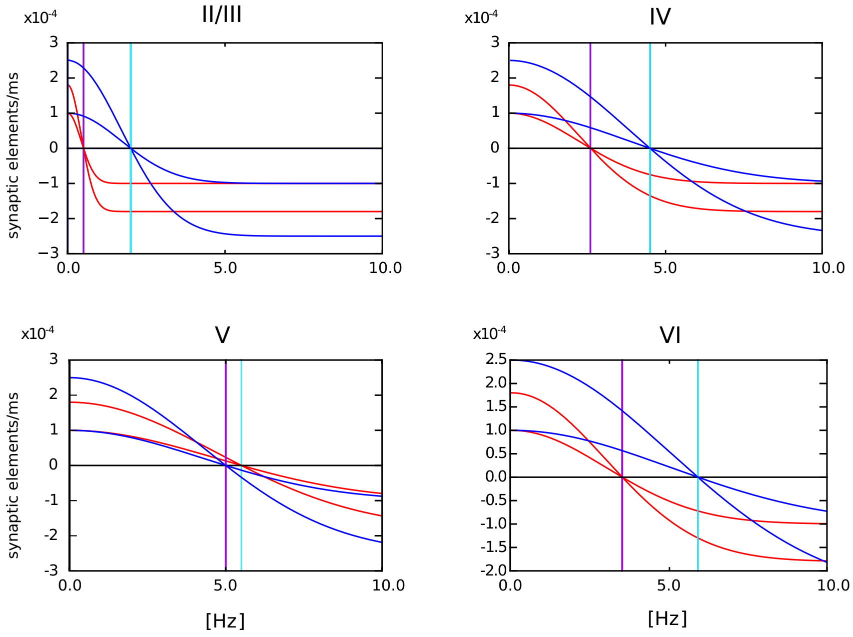


Figure 4.5: Growth curves for each synaptic element in each layer of the cortical microcircuit model. The growth curves define the rate at which synaptic elements are created depending on the firing rate of the cell. Red curves are for neurons in the excitatory population. Blue curves are for neurons in the inhibitory population. Solid lines are for the excitatory synaptic elements and dotted lines represent inhibitory synaptic elements. The vertical purple line defines the target firing rate for excitatory neurons and the vertical cyan line represents the target firing rate for inhibitory neurons. It is important to highlight that all synaptic elements of the same neuron must have a growth curve with the same target firing rate, otherwise equilibrium will never be reached.

connectivity. This means that the structural plasticity algorithm will stop creating and deleting synaptic connections when the desired mean activity is reached, and that this mean activity is actually reached on average in each population. In a second example, the growth rate provided leads to an unstable connectivity pattern, where the target mean electrical activities are never reached by all populations. A table containing the parameters for both cases can be seen in Table 4.2 on page 57.

Parameter	**Stable case**	**Unstable case**
Growth curve type	Gaussian	Gaussian
Growth rate excitatory dendritic	0.0001	0.0001
Growth rate excitatory axonal	0.00018	0.0001
Growth rate inhibitory dendritic	0.0001	0.0001
Growth rate inhibitory axonal	0.00025	0.0001
η excitatory	0.0	0.0
η inhibitory	0.0	0.0
ε in [Hz] - excitatory L23 dendritic	0.5	0.5
ε in [Hz] - inhibitory L23 dendritic	2.0	2.0
ε in [Hz] - excitatory L23 axonal	0.5	0.5
ε in [Hz] - inhibitory L23 axonal	2.0	2.0
ε in [Hz] - excitatory L4 dendritic	2.6	2.6
ε in [Hz] - inhibitory L4 dendritic	4.5	4.5
ε in [Hz] - excitatory L4 axonal	2.6	2.6
ε in [Hz] - inhibitory L4 axonal	4.5	4.5
ε in [Hz] - excitatory L5 dendritic	5.5	5.5
ε in [Hz] - inhibitory L5 dendritic	5.0	5.0
ε in [Hz] - excitatory L5 axonal	5.5	5.5
ε in [Hz] - inhibitory L5 axonal	5.0	5.0
ε in [Hz] - excitatory L6 dendritic	3.5	3.5
ε in [Hz] - inhibitory L6 dendritic	5.9	5.9
ε in [Hz] - excitatory L6 axonal	3.5	3.5
ε in [Hz] - inhibitory L6 axonal	5.9	5.9

Table 4.2: Parameters used for simulations of the cortical microcircuit. Values for both the stable and unstable cases are listed.

A third example was run to illustrate a more common situation where there are some assumption about the connectivity in a network and where the structural plasticity framework can help meet more refined activity constraints. Here I used the original model of Potjans and Diesmann [122] and enable the structural plasticity after an initial stabilization period of 30 s.

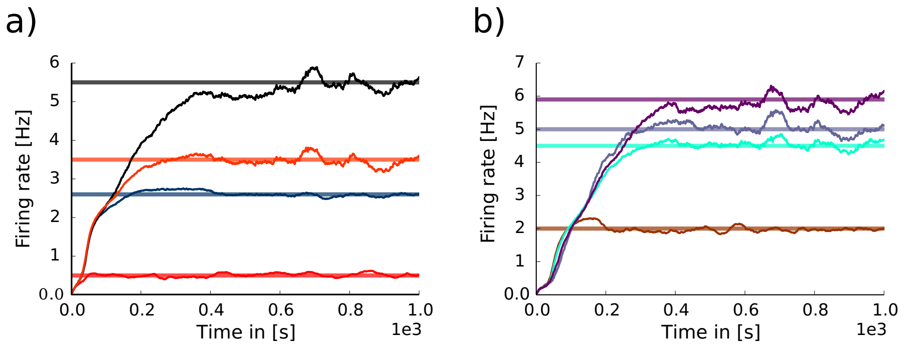


Figure 4.6: Evolution of the mean firing rate in each layer of the cortical microcircuit model. Pale horizontal lines indicate the target concentration of the corresponding population. A) excitatory populations in layers II/III (red), IV (blue), V (black) and VI (orange). B) inhibitory populations in layers II/III (brown), IV (cyan), V (gray) and VI (purple).

Simulations were performed on JUROPATEST (70 nodes with 2×14-core processors Intel(R) Xeon(R) CPU E5 $-$ 2695 v3 (Haswell) at 2.30 GHz and 128 GB DDR memory, running with Scientific Linux release 6.5) and JURECA (with 260 compute nodes with Intel Xeon E5 $-$ 2680 v3 Haswell CPUs with 2×12 cores per CPU, 128 GB of RAM per node and running on CentOS 7 Linux distribution).

In the case of the cortical microcircuit model, Figure 4.6 shows the changes in firing rate, while Figure 4.7 shows the evolution of connectivity among layers as the simulation runs. In this case, parameters which lead to stable network connectivity were chosen. Reaching stable connectivity in the networks takes around 700 biological seconds of simulation, which takes 24 hours using 25 nodes and 24 cores per node in the JURECA cluster to simulate. It is visible that during the first $20 - 30$ s of simulation, connectivity highly increases on every layer. After the initial overshoot, a smoother approximation towards the desired activity levels is achieved. As seen only from the firing rate diagram, the evolution of the network appears to be quite stable. Regardless, the connectivity plots show a continuous dynamical reorganization. While neurons on some layers might start deleting connections due to excess of activity, the post-synaptic neurons must then create new connections in order to compensate for missing activity in case they have not reached their set point yet. This leads to a continuous search for compensating excitation and inhibition which must satisfy the requirements of all 8 populations. From Figure 4.7 it can be seen that outgoing connections from excitatory populations on layers IV, V and VI are quite stable. On the

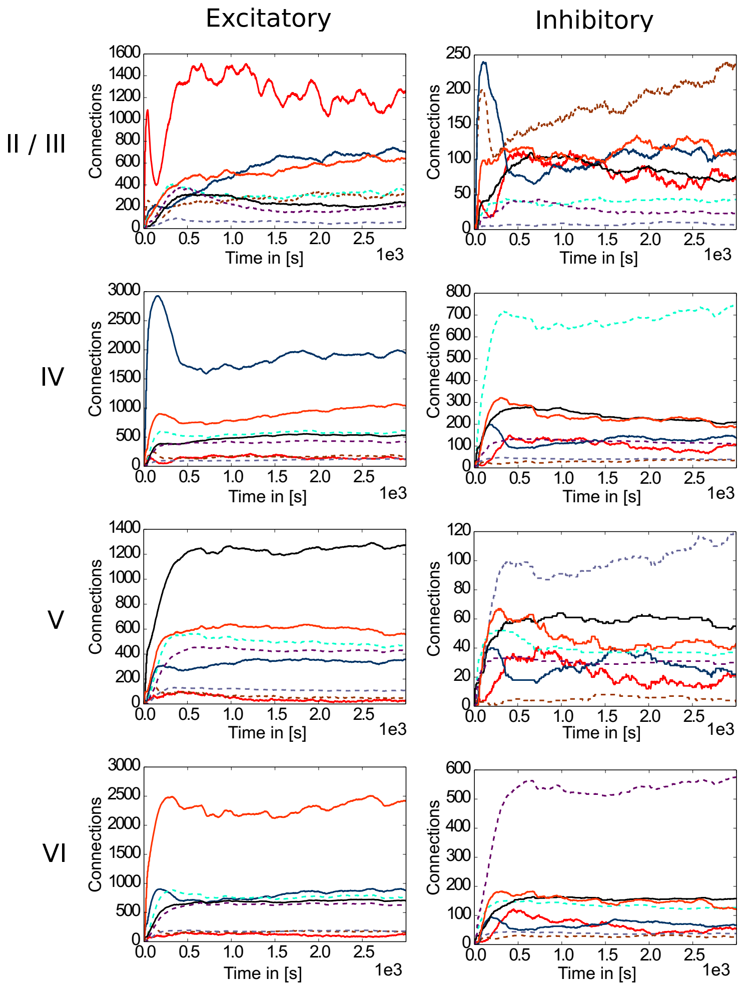


Figure 4.7: Evolution of connectivity in the microcircuit model resolved by source and target population. Panels on the left and right illustrate efferent connections from excitatory and inhibitory populations, respectively, while the vertical arrangement indicates the layer of the source neurons. In each panel, the numbers of connections to each of the eight population in the model are shown as a function of time. The colour of the curves indicates the target population, as in Figure 4.6: connections to excitatory populations in layers II/III (red), IV (blue), V (black) and VI (orange) are shown as solid curves, and connections to inhibitory populations in layers II/III (brown), IV (cyan), V (gray) and VI (purple) are shown as dashed curves.

other hand, layer II/III exhibits the highest amount of reorganization, both from the excitatory and inhibitory populations. This might be due to the fact that their reduced target levels of activity might be easily influenced by variations in all other layers. Inhibitory populations on all layers in general exhibit a higher degree of reorganization during the whole simulation.

The search space of connectivity parameters for this model of the cortical microcircuit is large as each setup requires 64 values to be defined. If a brute force exploration would be performed on these parameters by simulating each combination for 1 biological second, only $1-2$ values per parameter could be considered before more biological seconds would be simulated than using the structural plasticity approach. When adequate synaptic element growth curves are defined, the structural plasticity framework allows a progressive exploration of the space in which the dynamics of the the 8 populations are balanced at every step, thus providing an efficient way to find stable connectivity combinations.

Figure 4.8 presents a comparison between the proportional values of connectivity among layers between the results obtained from the simulation using structural plasticity and the original values reported by Potjans and Diesman. The average error in percentage of connectivity is of 1.058 ± 1.175.

A second case was also explored, in which parameters lead to unstable network activity are chosen. Figure 4.9 shows the evolution of connectivity among layers and Figure 4.10 shows the changes in firing rate in each layer for this scenario. Overshoots in the connectivity, are originated by a choice of higher rate in the creation of synaptic elements. The system behaves as a feedback control system, with a delay which is defined by the time between updates in connectivity and the synaptic element creation rate (see Chapter 3). The synaptic element growth rate determines the steepness of the growth curve, and influences the speed at which control changes are made. The instability in the connectivity is reflected in the firing rate, never reaching the desired levels. A stable setting involves finding a suitable balance between the speed in the creation of excitatory and inhibitory connections related to the desired level of activity for each layer.

In the use case where the initial conditions in connectivity were those specified in the original model of Potjans and Diesman, the network was simulated without plasticity for an initial period of 30 s in order to allow the firing rate reach an initial stable value. The evolution of the firing rate in all layers after the structural plasticity algorithm was enabled can be seen in Figure 4.11A and Figure 4.11B. The stability point is reached a lot faster than in the scenario with no initial connections, at around 400 s. A final simulation was set in which the connectivity was specified with a 10% error margin from the original setup reported by Potjans and Diesman. The

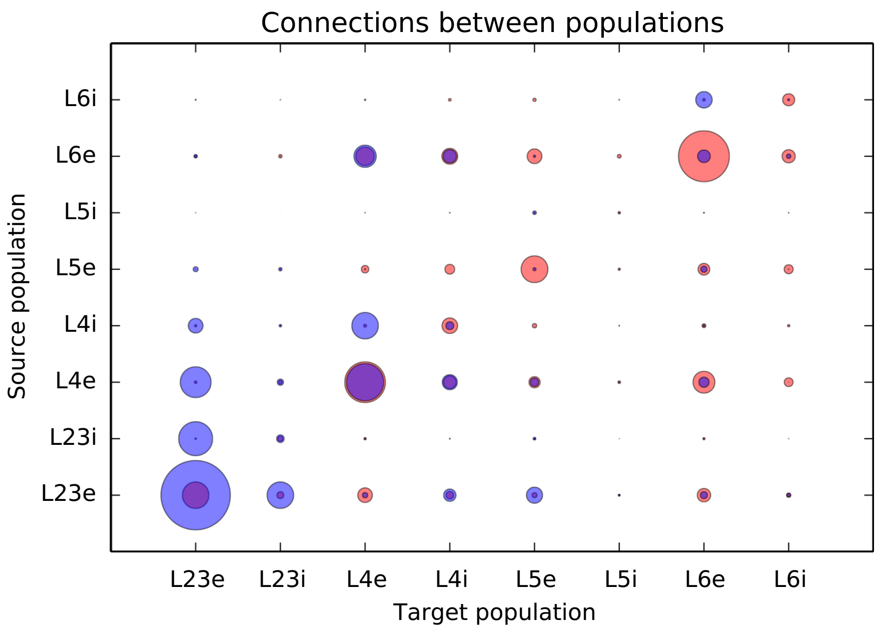


Figure 4.8: Comparison of the normalized connectivity in the microcircuit model between the results obtained with the structural plasticity framework (red) and the values reported by Potjans and Diesmann [122] (blue). The radius of the circle represents the linearly normalized value of the percentage of connections between layers.

evolution of the firing rate in all layers after plasticity was enabled can be seen in Figure 4.11C and Figure 4.11D. The structural plasticity algorithm is able to find a suitable balance between excitation and inhibition. The initial overshoot in electrical activity is a reflection of the initial stronger reconfigurations of the network connectivity. Not all setups will become stable or find a solution, this depends on the initial conditions, the desired set points, the shape of the growth curve and the growth rate.

4.1.3 Static visualization of simulation results

The lack of other restrictions in the generation or deletions of new connections may produce scenarios which are not biologically realistic. The emergence of biologically relevant networks is essential for its scientific applications and thus methods to monitor, control, correct or discard

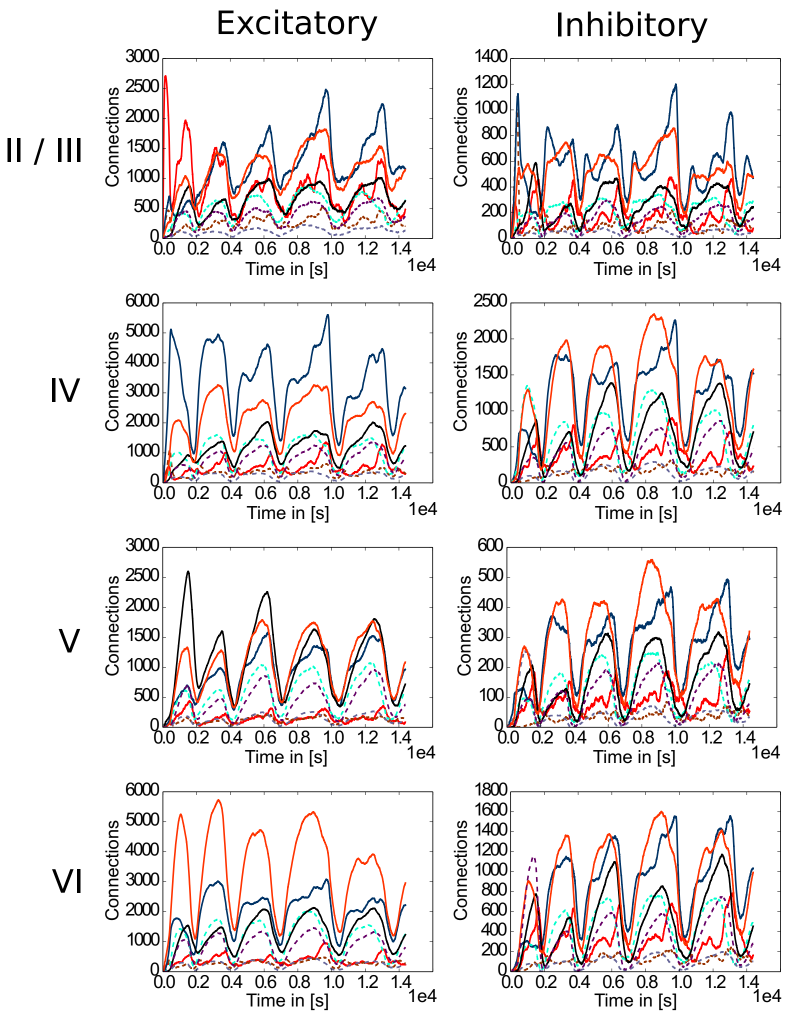


Figure 4.9: Evolution of connectivity through time for each layer in the cortical microcircuit model with an unstable set of parameters. Panels on the left illustrate connections incoming from excitatory populations. On the right, connections incoming from inhibitory populations. Rows show connections incoming from layers II/III, IV, V and VI from top to bottom respectively. On every panel, red solid lines target the II/III excitatory population, brown dotted lines target the II/III inhibitory population, blue solid lines target the IV excitatory population, cyan dotted lines target the IV inhibitory population, black solid lines target the V excitatory population, gray dotted lines target the V inhibitory population, orange solid lines target the VI excitatory population and purple dotted lines target the VI inhibitory population.

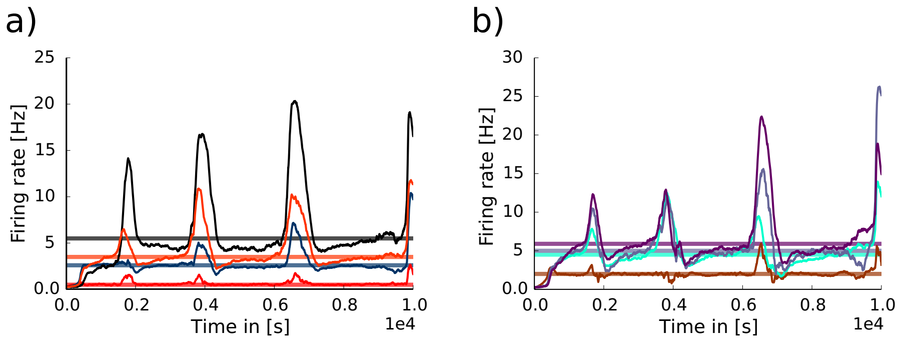


Figure 4.10: Evolution of firing rate in each layer of the cortical microcircuit model with an unstable set of parameters. A) shows the firing rate in the excitatory populations. Red curve represents layer II/III, blue curve layer IV, black layer V and orange layer VI. B) shows the firing rate in the inhibitory populations. Brown curve represents layer II/III, cyan layer IV, gray layer V and purple layer VI.

unrealistic configurations are of essence. MSPViz [21] is a software tool developed to analyze post-simulation data of the progressive connectivity generation in the network. Within the framework of my thesis, I defined the requirements and use cases for the development of this visualization tool. With it, the scientist can explore the network at detail as well as the generation and deletion of specific synapses. The user can look for abnormal connectivity patterns such as nonfunctional hubs or detect interesting behavior as the emergence of critical periods. The scientist can identify the origin of these patterns by inspecting the dynamics of the inhibitory/excitatory synaptic element creation/deletion.

Identifying and analyzing critical periods of plasticity

Identification of critical periods of plasticity, its study and analysis is one of the most important topics for structural plasticity. A critical period is a phenomenon where strong rewiring, creation and deletion of synapses take place in a neural network. Critical periods are tightly linked to learning and healing and seem to be more frequent during development [69]. These periods of increased plasticity can be triggered by inhibition imbalances in a network. In order to illustrate the usability of MSPViz, I have chosen the simple model with two populations of point neurons with a target average activity of the inhibitory population is set to 20 Hz while the target average activity in the excitatory population is set to 5 Hz.
Using MSPViz the user can load, explore, analyze and understand the data generated by NEST

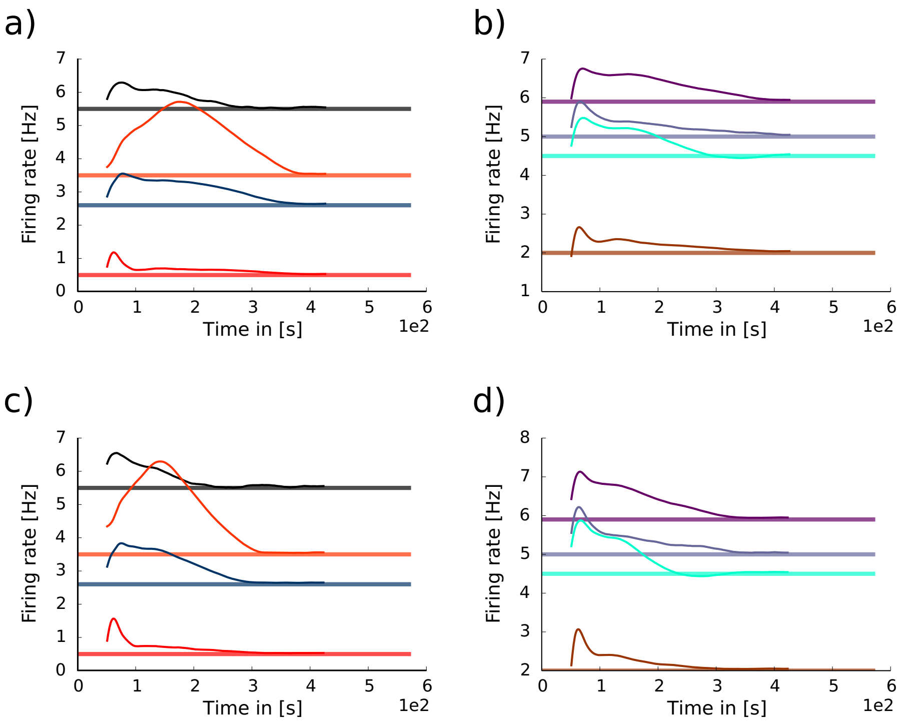


Figure 4.11: Evolution of firing rate in each layer of the cortical microcircuit model with partially pre-connected initial conditions. Pale horizontal lines indicate the target concentration of the corresponding population. Left panels show excitatory populations in layers II/III (red), IV (blue), V (black) and VI (orange). Right panels show inhibitory populations in layers II/III (brown), IV (cyan), V (gray) and VI (purple). A) and B) show the scenario where the network was started with the connectivity as specified in the original work by Potjans and Diesman. C) and D) show the same scenario but with a 10% error in the initial connectivity setup.

simulations using structural plasticity. The data output by the simulation describes the existing connections in the network at every 1000 steps of simulation. It also contains information about the available, empty and used synaptic elements of each type in each neuron. MSPViz is designed to allow a thorough inspection of the changes in the synaptic elements of each neuron as well as the synapses it forms with other neurons. This is done by using an abstract representation of the morphology of each neuron. For an example, see Figure 4.12.

Once the software has loaded the simulation data, the user can navigate through it using different views. A generic view of the network allows easy inspection of the dynamics of the connectivity.

The user can ”play” the simulation and observe how connections are created and deleted among neurons (see Figure 4.13). There is the option to select just a group of neurons and make more detailed analysis. In this use case, the user can also observe how the network goes through periods of critical plasticity which impact its final structure. The simulation shows how inhibition triggers a rewiring of the network around step 35. It is also possible to identify the particular point at which excitatory to excitatory connections start to grow again after the initial settling period (1).

Identifying hubs

Nonfunctional hubs are neurons with a large number of connections but a very low firing rate. This means that they do not take active part in the dynamics of the network. These hubs are undesirable in a biologically realistic setup. They can emerge when the homeostatic rules set for excitatory and inhibitory synaptic elements are not well defined. Using MSPViz, the user can select the nonfunctional hubs, visually identified by a high number of connections and a low activity, for further inspection as seen in Figure 4.14. We can see that, for this neuron, starting cycle 35 (where the critical period of rewiring in the network occurs) a inhibitory/excitatory race starts. Here, excitation is matched and slightly overtaken by inhibition. This leads to more synaptic elements being created of both types and, as a consequence, firing rate goes down until the neuron turns into a non functional hub. Figure 4.15 shows an even more detailed view on a nonfunctional hub neuron with the dynamics of its connections and allows the exploration of the network using the synaptic map. The user can clearly identify the cascade of events which lead to the loss of functionality of the neuron. The user can also use the visible connections to explore the effects of these changes on neighboring neurons.

4.1.4 Taking into account distance dependency in simulations

The implementation described in this chapter neglects the distance between neurons as a factor for deciding the targets during the generation of new connections. This is an important feature highlighted in Chapter 2, Section 2.3.2 and also part of the original model of structural plasticity suggested in Butz and van Ooyen [22]. Calculating the distance between all possible targets and adding this information to the target selection is a computationally intensive process. The calculations increase quadratically with the number of neurons. There have been some attempts to solve this problem. One is explained in Rinke et al. [125], where the authors propose the progressive segmentation of the space into squares, and the distance calculations only performed

on the centroids of such partitions. This changes the complexity of the problem from $O(N^2)$ to $O(NlogN)$. Other approaches, like the one explained in Bogdan et al. [15], reduce the potential space of targets *a priori*, assuming that long distance synapses will not be possible.
In this section I want to discuss another way to take distance into consideration. This way of dealing with the problem can be achieved in the current implementation of NEST. This solution is only applicable for networks without explicit spatial position, such as the majority of example models in NEST, among which the two networks discussed in this chapter. In NEST, distance can be linked directly to the delay of the synapse, and thus one can create sets of plastic synapses with different delays. The modeler then can assign slightly higher growth rates to synaptic elements linked to faster synapses. By doing this, 'closer' connections will be proportionally favored to long distance connections.
This approach has its limitations, in that there should be consistency in the delays of the connections between two neurons. Making sure that a neuron can not have connections with two different delays to the same neuron would promote this consistency from bottom up. The development of an algorithm which controls this spatial consistency within the network could enable a low complexity solution for self-generating networks with spatial distribution as well.

4.1.5 Changes in the implementation of structural plasticity for the 5th generation kernel of NEST

As the NEST kernel changed from 2.10 up to 2.16 there were no substantial changes in the implementation of structural plasticity. However the 5th generation kernel (5G) [78] included a radical modification in the way connections are implemented. Communication of spiking events in the 5G kernel is based on an AllToAll communication scheme instead of an AllGather strategy as the previous kernel. This means that events are exchanged between processes in a directed way, reducing the transmission of useless information to non interested targets. The new communication strategy also involved a change in the definition of synapses. In the previous versions of NEST, synaptic information was only stored in the post-synaptic neuron but in the 5G kernel both the pre- and post-synaptic neurons contain information about the synapse. This allows the directed exchange of event information to take place. Synaptic information in 5G is allocated in the post-synaptic process as two resizable three dimensional structures which hold information about the synapses and about the sources. This information is indexed using the target thread id and the type of synapse. On the pre-synaptic process, there is a three dimensional

structure which holds the information about the targets indexed by source thread and source local id. For more details on the implementation please refer to section ”3.1. Two-Tier Connection Infrastructure” of [78]. The process of building up connectivity in 5G is performed in two steps. The first step generates all the post-synaptic data infrastructure. The second step takes place at the beginning of the simulation and includes the transmission of synapse data from the targets to the sources in order to generate the pre-synaptic connectivity data structures.
The most important change introduced by 5G regarding structural plasticity is the duplication of information between both sides, and the fact that synapses are ordered by source in the post-synaptic data structures. This means that any removal or addition of synapses involves reordering the data structure in order to preserve the right pointers towards sources. The current version of 5G includes the creation and deletion of synapses during simulation and is able to reach a slightly better performance than 4G in networks with less than a 100 MPI processes. The impact of regenerating the corresponding communication data structures after the creation or deletion of synapses can be hidden by a low frequency of updates in the connectivity. This means that just like with 4G, 5G requires large update intervals compared to the simulation time step in order to have good performance. Its scalability to higher numbers of processes is still work in progress. The new communication scheme in 5G opens up the possibility to implement several changes in the original algorithm in order to make faster transmission of the synaptic element data among processes which is currently a time consuming task within the algorithm.

4.2 Discussion

In this chapter I have described the implementation of a framework of structural plasticity for the neural network simulator NEST. As shown in [37], the framework is scalable and can be used to model the dynamical creation and deletion of synapses inside a large scale network guided by simple homeostatic rules. This work also presents some use cases for the framework and some of its potential applications. Researchers can now use structural plasticity in NEST to generate the connectivity of a network from scratch, defining homeostatic rules, in form of synaptic element growth curves, which may vary according to their needs. Some examples of research work done using this framework are Gallinaro and Rotter [54] and Lu et al. [98]. The shape of the growth curve defines the speed with which new synaptic elements are created, and as a result, defines the acceleration at which calcium is stored inside the neuron. The relationship between the growth speed at certain firing rate of excitatory and inhibitory elements is fundamental to achieve

stable setups under the model of structural plasticity. As is has been shown, some parameter combinations lead to unstable activity in the network. There are cases where the desired average electrical activity will never be reached by the system. In other cases the average electrical activity will oscillate continuously or suddenly go out of bounds. This relationship depends also on the size of the network and the neuron model used. As a consequence, some care is required in navigating the parameter space in order to achieve desired results.
The example of the two population network illustrates how this framework can be used to understand the interaction between activity and the creation of synapses. The behaviour observed in the simulation can be used to model how inhibition triggers critical periods of connectivity during development of neural networks [68]. During this window, external stimuli can also be used to shape the formation of the new connections. Together with the performance measurements, these results show that this implementation of structural plasticity is suitable to study the development of connectivity patterns inside a neural network in an efficient and scalable manner.
In the specific case of the cortical microcircuit presented in Diaz-Pier et al. [37] it is possible to see some similarities and differences between the results obtained by simulating with the structural plasticity framework and the data reported in Potjans and Diesmann [122]. One of the most visible differences is the smaller amount of recurrent connections generated in the simulation for layer 2/3. This layer has a very low target electrical activity, which is initially almost reached by external input. This means that very few synapses are required to reach this target. This fact limits the creation of synapses for this layer. Note that the results shown in this chapter were obtained only by defining target activity levels; no other connectivity constraints were specified. A more elaborate simulation could incorporate tailored growth curves for each layer, and implement additional connectivity restrictions which promote recurrent connections and other connectivity patterns that do not emerge naturally from the current approach.
Another visible difference is that the excitatory population of layers 5 and 6 show a higher number of connections than the ones shown in the original work. On the other hand, connections from and to the inhibitory population of layer 5 and layer 2/3 are well fit. Except from connections between the inhibitory and excitatory populations of layer 4, connections from and to layer 4 are also well predicted.
In this work I describe the implementation of a framework which can be used to study structural network dynamics. The focus of this work is on the technical implementation. It was not the scope of this chapter to perform a deep analysis of the biological results that can be obtained using this framework. However, some examples of its applications are used to highlight its capacities

and limitations. For more complex applications a further analysis please refer to Chapter 6 and Chapter 7. The structural plasticity framework gives researchers flexibility to explore complex connectivity dynamics by extending the synaptic elements growth rules. As this implementation is integrated into NEST, simulations using structural plasticity can also be combined with other features available in the simulator. For example, the user may take into account dynamic synaptic weights by mixing this framework with synaptic plasticity. The framework can also be further extended using the current topology framework in NEST in order to constrain connectivity by relative position.
In this section I also showed how the MSPViz software can be used to explore the changes in the network produced during a simulation with structural plasticity. MSPViz provides means to analyze the information produced by the simulations and assess if the parameters used are producing biologically relevant information. It also allows the user to better understand the relationships between the activity of the neurons and the impact of changes on an abstract representation of their morphology.
The structural plasticity algorithm is able to solve the complex balance of interaction between layers with different levels of electrical activity when partial information of the connectivity is available. This result is very promising, as it shows that given the right growth rules, it would now be possible to reconstruct connectivity inside a network without having exact anatomical information. As a conclusion, this approach represents a novel and useful technique to close the current gaps in information about the connectivity in certain regions of the brain.

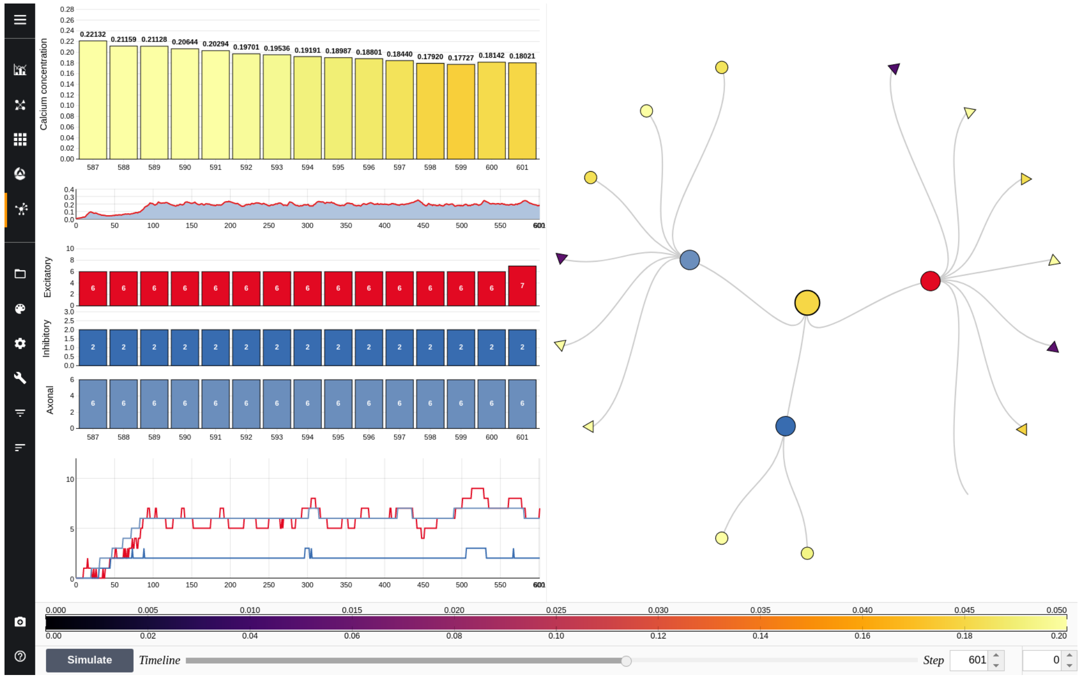


Figure 4.12: Visualization of the abstract ”morphology” of individual neurons. Focus here is on an inhibitory neuron. Top left: bar chart and plot of the firing rate of the neuron during the simulation. Bottom left: bar chart and plot of the number of synaptic elements for this neuron (dendritic excitatory and inhibitory as well as axonal inhibitory). Right: diagram of the neuron’s ”morphology” and its synaptic contacts. By selecting other connected neurons the right panel, the user can navigate through the network.

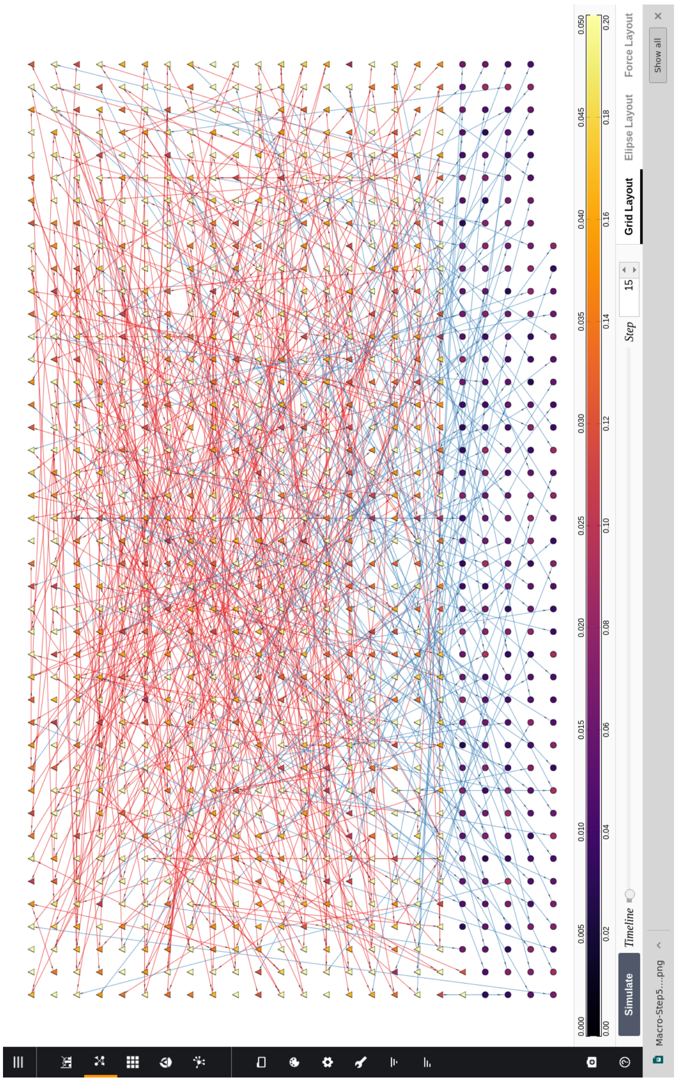
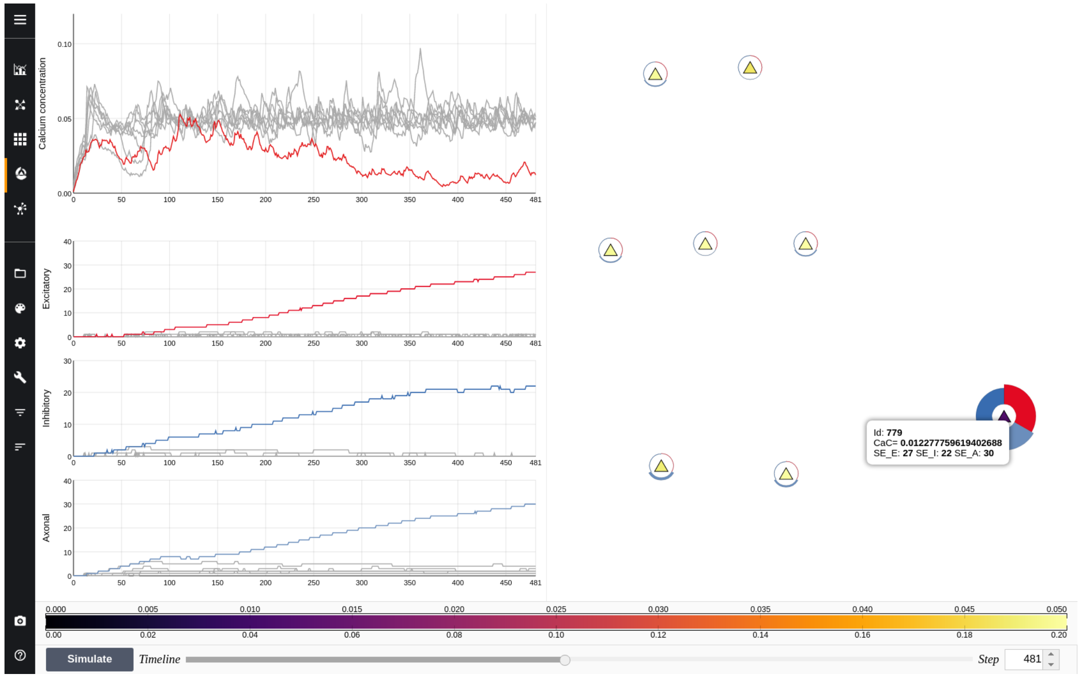


Figure 4.13: General view of the network. Excitatory and inhibitory neurons are represented as triangles and circles respectively. Neurons are displayed on a grid. Excitatory and inhibitory synapses are by default red and blue respectively. Colors can be changed by the user in the setup menu. The color bar in the bottom shows the colors and scale for the firing rate of the neurons. The time line in the bottom shows the evolution of the simulation and can be used to select specific points in time.



Id: **779**
CaC= **0.012277759619402688**
SE_E: **27** SE_I: **22** SE_A: **30**

Figure 4.14: Selection of a group of neurons including a non functional hub. In this view, the information about each type of connections in the neuron is synthesized in a circular graph around the neuron. Each portion of the circle represents a type of synapse. The size of the portion indicates the amount of connections of each type. This allows a quick inspection of the relationship between each synapse type in the neuron.

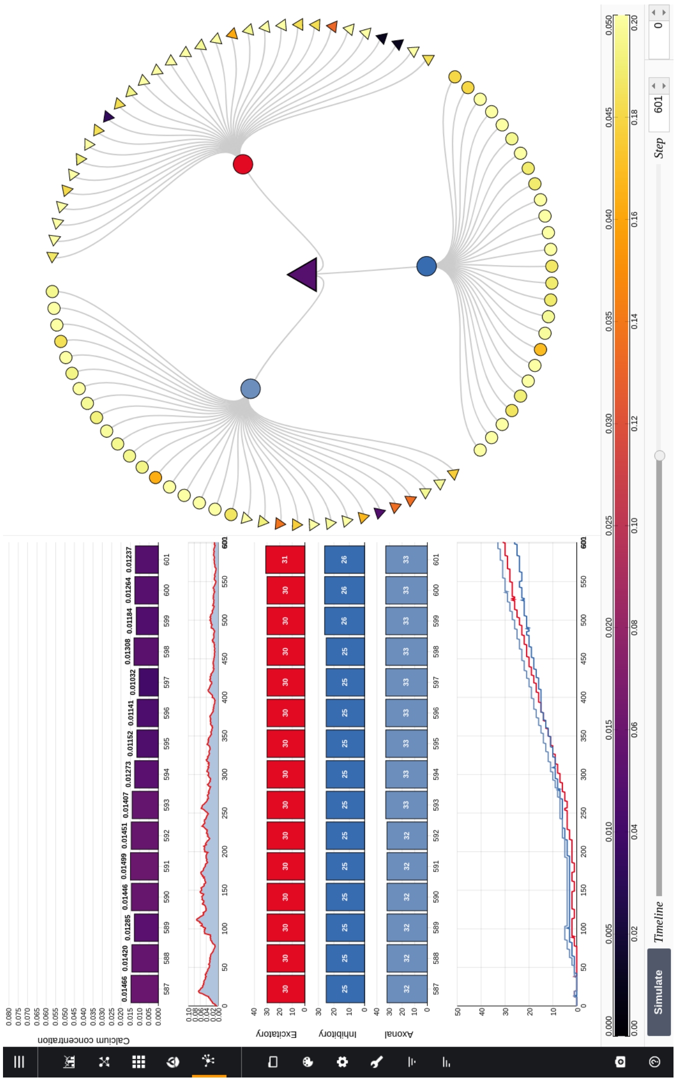


Figure 4.15: View of a neuron which has turned into a non functional hub. It can be seen how the firing rate of the neuron has progressively decreased during simulation time. In contrast, the number of connections and synaptic elements have increased. The lower left plot shows how the inhibitory and excitatory connections race between each other, balancing out and generating the non functional hub.

5 Steering and interactive visualization of structural plasticity

In the last chapter I discussed the characteristics of structural plasticity as a controller of neural networks by means of structural changes. In this chapter, I focus particularly on connectivity generation using structural plasticity in combination with interactive steering and visualization. Simulation models in many scientific fields can have non-unique solutions or unique solutions which can be difficult to find. Moreover, in evolving systems, unique final state solutions can be reached by multiple different trajectories. Neuroscience is no exception. Often, neural network models are subject to parameter fitting to obtain desirable output comparable to experimental data. Parameter fitting without sufficient constraints and a systematic exploration of the possible solution space can lead to conclusions valid only around local minima or around non-minima. Finding suitable connectivity configurations for neural network models constitutes a complex parameter search scenario as well as a relevant and still unsolved problem in computational neuroscience.

In Nowke et al. [114] we developed an interactive tool for visualizing and steering parameters in neural network simulation models. The development of the tool has been guided by several use cases – the tool allows researchers to steer the parameters of the connectivity generation during the simulation, thus quickly growing networks composed of multiple populations with a targeted mean activity. The flexibility of the software allows scientists to explore other connectivity and neuron variables apart from the ones presented as use cases. With this tool, we enable an interactive exploration of parameter spaces and a better understanding of neural network models and grapple with the crucial problem of non-unique network solutions and trajectories. In addition, we observed a reduction in turn around times for the assessment of these models, due to interactive visualization while the simulation is computed.

5.1 Exploration of the parameter space of neural network connectivity

Neuronal models and neural mass models, usually based on coupled systems of differential equations, contain many degrees of freedom which determine the dynamics of the system. In a neural network, these models are interconnected and the strength of the interactions between elements can also change through time.

Since biological evidence to specify a complete set of parameters for a neural network model is often incomplete, conflicting, or measured to an insufficient level of certainty, parameter fitting is typically required to obtain outputs comparable to experimental results (see, for example, López-Cuevas et al. 97, Schuecker et al. 134, Zaytsev et al. 168, Schirner et al. 133). And even *if* we had infinite experimental data available, Cubitt et al. [30] have shown that, regardless of how much experimental data is acquired for a general system, the inverse problem of extracting dynamical equations from experimental data is intractable: "extracting dynamical equations from experimental data is NP hard". This implies that in neural networks, the problem of finding the exact free parameters for a simulation leading to results matching experimental measurements cannot be solved in polynomial time, at least under the current understanding of computational complexity.

However, the parameter space can be explored with forward simulations in order to discover the system's characteristic behaviors and thus limit the search space to a computationally tractable sub-problem in an educated manner. The definition of these subspaces can then be the basis for robust – and non-arbitrary – parameter determination (in other words, mathematically valid performance function minimization). In fact, given the known mathematical characteristics of the dynamics of neuronal and neural mass networks, investigators should characterize the solution spaces of sufficiently complex networks and models before selecting what they propose are statistically diagnostic simulation trajectories. In practice, this rarely happens, even though parameter fitting without sufficient constraints and a rigorous exploration of the possible solution space can lead to conclusions valid only around local minima or around non-minima. Researchers frequently stay within arbitrary regions in the parameter space which show interesting behaviors, leaving other regions unexplored.

Visual parameter space exploration has been successfully applied in several key scientific areas, as detailed by Sedlmair et al. [136]. Combined with interactive simulation steering, the time for obtaining optimal parameter space solutions can be significantly reduced [104, 105]. Whitlock et al. [161] present an integration of VisIt [27], a flexible end-user visualization system, into existing simulation codes. This approach enables in situ processing of large datasets while adding visual analysis capabilities at simulation runtime. A similar approach has been suggested by Fabian et al. [53] for ParaView [65].

Coordinated multiple views (CMVs) as proposed by North and Shneiderman [110] and Wang Baldonado et al. [157] can assist in visual parameter space exploration. CMVs are a category of visualization systems that use two or more distinct views to support the investigation of a single conceptual entity. For example, a CMV system can display a 3D rendering of a building (the conceptual entity) alongside a top-down view of its schematics — whenever a room is selected within the schematic overview, the 3D rendering will highlight the room's location. Roberts [126] shows that CMVs support exploratory data analysis by offering interaction with representations of the same data while emphasizing different details. Ryu et al. [129] present CMV systems that have been successfully utilized to uncover complex relationships by enabling users to relate different data modalities and scales, and assisting researchers in context switches, comparative tasks, and supplementary analysis techniques. Additional examples of such systems are presented by North and Shneiderman [111], Boukhelifa and Rodgers [19], and Weaver [158].

Visual exploration of neural network connectivity, e.g., by displaying spatial connectivity data in 3D renderings, has previously been employed by scientists to better understand and validate models as well as to support theories regarding the networks' topological organization [108, 127]. The infinite solution space of suitable connectivity paths and end configurations for neural networks makes fully automatic parameter fitting 'hard', since it involves satisfying multiple contradictory objectives and qualitative assessment of complex data, as explained by Sedlmair et al. [136]. Kammara et al. [79] conclude that for multi-objective optimization problems, visualization of the optimization space and trajectories permits more efficient and transparent human supervision of optimization process properties, e.g., diversity and neighborhood relations of solution qualities. They also point their work toward interactive exploration of complex spaces which allows expert knowledge and intuition to quickly explore suitable locations in the parameter space.

In order to address efficient but rigorous parameter space exploration, in Nowke et al. [114] we have reported the development of an interactive tool for visualizing and steering parameters in neural network simulation models. In this work, we focused particularly on the generation of connectivity, since finding suitable connectivity configurations for neural network models constitutes a complex parameter search scenario. The generation of local connectivity is achieved using structural plasticity in NEST [17] following simple homeostatic rules described in Butz and van Ooyen [24]. The concepts presented in Chapter 3 are used to specify the problem from the control theory perspective, as variations in the structure system control the transition in its dynamics from an initial to a final state following a defined trajectory. The tool allows researchers to steer the parameters of the structural plasticity during the simulation, thus quickly growing networks composed of multiple populations with individually targeted mean activities. The flexibility of the software allows the exploration of other connectivity and neuron variables apart from those presented as use cases. CMVs are used to interactively plot firing rates and connectivity properties of populations while the simulation is performed. Moreover, simulation steering is realized by providing interactive capabilities to influence simulation parameters on the fly.

The development of this tool was guided by two use cases where visual exploration is key for obtaining insights into non-unique dynamics and solutions. The first use case focuses on the generation of connectivity in a simple two population network. Here the generation of connectivity to achieve a desired level of average activity in the network can be achieved by taking multiple trajectories with different biological significance. The second use case is inspired by a whole brain simulation described in Deco et al. [35], where the exploration of non-unique connectivity solutions is desired to understand the behavior of the model.

Applying this approach, an intractable inverse problem can be reduced to a tractable subspace, and the requirements for statistically valid analyses can be determined. Visualization can simplify a complex parameter search scenario, helping in the development of mathematically robust descriptions amenable to further automated investigation of characteristic solution ensembles. Observing the evolution of connectivity, especially in cases where several biologically meaningful paths may lead to the same solutions, can be useful for a better understanding of development, learning and brain repair. This work is a first step toward developing new analytic and computational solutions to specific inverse problems in neuronal and neural mass networks. Our software platform promotes rigorous analysis of complex network models and supports well-informed selection of parameters for simulation.

5.2 Connectivity generation in neural networks

Previous research by Sporns et al. [139] has found that the assembly of anatomical connections among neurons, also known as the connectome, plays a fundamental role in explaining the high-level activities of the brain. However, the exact relationship between anatomical links and the functions performed by the brain has aspects that remains unclear. An attempt to model biologically realistic circuits immediately runs into the problem that the structure of the brain has yet to be comprehensively characterized. Existing connectomic datasets are incomplete or contain large uncertainties [10]. Conversely, information about the average electrical activity in specific brain regions is easier to acquire either directly, e.g., electroencephalogram, extracellular electrode recordings of spiking activity and local field potential, or indirectly, e.g., functional magnetic resonance imaging and optogenetics/calcium imaging.

Variations in the physical elements, which constitute a neural network, can be modeled using synaptic and structural plasticity. Structural plasticity, a model of the dynamic creation and deletion of synapses in a neural network, is desirable from two main perspectives. The primary purpose is to study the neurobiological phenomenon of morphological transformations that a neuron or set of neurons undergoes through time, leading to the creation or deletion of synapses. This phenomenon is part of brain development, learning and repair. However, a promising secondary role already introduced in Chapter 4 is the automatic generation of neuron-to-neuron synapses to compensate for gaps in experimental connectivity data. Using structural plasticity, a network can autonomously generate synapses to achieve a stable desired profile of electrical activity, a measure that is experimentally more accessible than detailed connectivity data. By progressively and slowly changing the connections between neurons in the network and the weight of these connections for all regions, the structural plasticity algorithm is able to find stable configurations within the desired firing rate profile.

As alreday mentioned in Chapter 4, the structural plasticity implementation in NEST is based on the model proposed by Butz and van Ooyen [24] and described in detail in Diaz-Pier et al. [37]. In this Chapter I use a Gaussian curve as described in Equation (2.2) and the properties of the plasticity model as described in Section 2.3.1. The original model by Butz and van Ooyen [24] uses intracellular calcium concentration as a proxy for the mean firing rate. The examples included in this Chapter reference directly the mean firing rate in the homeostatic rule. In the simulations shown in this chapter, the form of this curve is not biologically motivated, but is a homeostatic meta-rule being used to numerically solve for networks consistent with fixed firing rates.

As already explained in Chapter 4, the firing rate is calculated by low-pass filtering spike train data by convolving that data with an exponential decay kernel. This calculation is internal to NEST and independent of the tool discussed in this chapter. When the convolution technique isn't suitable, an alternate mean firing rate can be computed using a user-defined window size applied to binned spike trains.

As discussed in Chapter 3, synaptic and structural connectivity can be seen as multi-objective optimization algorithms which take the network from an initial state to a final state where *something has been learned* or *a new activity pattern has been enabled*. Partial information about the connectivity can be combined with information about average activity in the system to initialize models of structural plasticity filling the gaps in the constraints of the system. However, finding suitable connectivity configurations and generation trajectories for neural network models is non-trivial, which is exacerbated by the nature of experimental data. The known experimental data often fails to sufficiently constrain the model to parameter subspaces that can be completely explored with reasonable resources within reasonable time frames.

Enabling structural plasticity for a single population to reach a targeted activity level is usually unproblematic, fast, and relatively insensitive to the choice of parameters such as ν and η. However, a big challenge arises when structural plasticity is involved simultaneously on several interconnected populations with differing levels of activity. Even small changes in the connectivity of each population will impact the activity of all others to which it is connected, leading to a propagated destabilization. Another parameter which has a great impact on stability is the update interval at which synapses can be deleted or created. As in any control system, the delay between a control change and the response of the system strongly determines the capability of the controller to keep the system in a stable region.

In Diaz-Pier et al. [37], the simulations were performed statically, meaning no steering was possible during runtime. Due to the large combination of parameters to be controlled and variables to be observed during the search process, brute-force parameter search based on static simulation proved to be insufficient to obtain stable states. The selection of adequate parameters to define and constrain the growth of network connectivity, especially for multi-population or coupled networks, is not trivial as some values might lead to unstable setups. Therefore, modifying the characteristics of the growth behavior (ν and η see Figure 2.2) for each population and the update interval *during simulation* becomes crucial for finding a suitable stable state for multi-population networks. In this chapter the terms ”population” and ”region” are used interchangeably to refer to groups of neurons. The term chosen depends on the use case. In general, a region contains one or more populations while populations specify groups of neurons of the same type. Connectivity exists both within and between populations and regions. All types of connectivity can be subject to plasticity or remain fixed after setup. The software can be modified to take into account any number of populations per region, arbitrary types of neurons, and any number of regions. The user can also specify different types of connections between the same populations and apply various structural plasticity rules to each of them. The user can choose between a variety of connectivity modalities in NEST, ranging from one-to-one, all-to-all, fixed in-degree, fixed out-degree, fixed total number of connections, and pairwise Bernoulli. However, structural plasticity support is only currently implemented for one-to-one and all-to-all connectivity. Other modalities can be used, but structural plasticity will not affect these connections.

In the context of a simulation with evolving connectivity, the dynamic nature of the parameter search workflow derived from the two use cases presented later requires:

W1: The simultaneous analysis of several changing variables by an expert.

W2: Comparing the level of activity of several populations simultaneously.

W3: Changing simulation parameters at any moment in each population of the network.

W4: Snapshotting a time point in the simulation and storing the connectivity state.

W5: Loading a previously stored connectivity state.

This workflow can potentially be assisted with an interactive tool enabling scientists to explore and steer such simulations within the space of possible trajectories. To achieve this goal, a scientist needs interactive feedback on the number of connections and the level of electrical activity in all populations.

More specifically, the initial design phase helped identify the following visualization requirements (**R1-R5**), followed by the requirements for simulation steering (**R6-R10**). These requirements hold for the presented use cases:

R1: Deal with at least $2 \times N$ representations of time series data (electrical activity and connectivity), where N is the number of populations in the simulation.

R2: Interactively plot the firing rate for selected populations. The firing rate from the last simulation step should be displayed as soon as its computation concludes.

R3: Interactively plot connections for each population. As for the firing rate, the latest total connections per population should be displayed.

R4: Enable the selection and filtering of populations for plotting and further investigation. The means to select and filter populations of interest must be provided.

R5: Have a well defined way to distinguish populations in the plot. Since multiple populations can be selected for comparison, visual clutter needs to be avoided.

R6: The user interface must allow for the modification of each population's growth rate ν and apply each value in the simulation.

R7: The user interface must allow for the modification of a population's minimum electrical activity η and transfer the new value to the simulation engine.

R8: The user interface must allow for the modification of the update interval and transfer its change to the simulator.

R9: Control the NEST simulation from within a graphical user interface. Provide the means to start or stop the simulation, trigger the saving and loading of a network state, and allow convenient access to the visualizations.

R10: Enable loading and saving of the current network state (connections and user controlled parameters).

5.3 In situ visualization and steering of connectivity generation

To enable navigation through the connectivity generation parameter space, in Nowke et al. [114] we developed a tool enabling interactive steering and visualization (ISV). The development was driven by the need to rapidly reach stable configurations of connectivity in multiple tightly connected populations. We then extended the tool to support further use cases which are presented later. The tool allows for the visualization of trajectories that the system undergoes during simulation by showing the changes in the observable states of the network (specifically the activity and connection properties of the network). In addition, this tool allows for the modification of the control signals for the generation of connectivity, i.e., the plasticity algorithm's parameters.
The developed tool realizes a CMV system by applying principles of event-driven architectures as presented in Abram and Treinish [3], Michelson [107], and Nowke et al. [112]. The development of the tool was organized into four stages: first, the simulation script was modified to

retrieve electrical activity and connectivity values; second, the visualization components and user interfaces were developed; third, processing of parameter changes from the user interface was added; and finally, the simulation script was optimized to run on supercomputers.
The tool works by the combination of two components: a visualization framework and a simulation instrumentation API. Event-communication is realized with the '*nett*' messaging framework [113], which is an open source C++ network library facilitating data transfer between application boundaries based on the publish and subscribe pattern.

5.3.1 Simulation instrumentation

Interactive steering relies on a bidirectional communication between the visualization and steering interfaces to a simulator. In our setup, activity levels and connectivity from populations computed by NEST are transferred via event communication over a network connection to the visualizations, where users can modify parameters of the simulation model, which in turn are fed back to the simulator. The values of interest are the firing rate of each population which serves as a proxy for electrical activity and a population's total connections formed due to connectivity generation. These are the observable states of the network. Steering parameters are the minimum firing rate η and the growth rate ν of each population, the update interval for the connectivity generation, and finally, basic commands to NEST such as ending or resetting the simulation, and storing or loading the current network state.
To retrieve firing rates and total connections, instrumentation of the simulation script is required. To this end, the simulation acquires the latest firing rates and total connections of each population in each iteration and publishes these as events. Then, parameter changes from the graphical steering interfaces, asynchronously retrieved during the model's computation, are applied and the next iteration is continued.
To adapt a NEST simulation to a different use case, the first step consists of determining what data needs to be transferred from or to the simulation. The next step consists of creating an event definition schema for the data to be transferred if one is not yet present. Then, slots for communicating this data definition can be created: out-slots for publishing data and in-slots to retrieve it. Once slots are created, in-slots need to be connected to their corresponding out-slots. Any in-slot should be used in a thread to asynchronously retrieve data without blocking the computation of the simulation. Once an event is received by a slot, its data needs to be applied in the next iteration of the simulation. In a complementary fashion, out-slots send the simulation

results for each iteration by retrieving values of interest from the simulation and filling the slot's event and sending it. The same methodology is used for visualizations or graphical user interfaces which are use case specific.

5.3.2 ISV system overview

The ISV system consists of six services:

1. Control panel: It serves as an entry point for users to start the investigation of structural plasticity. The user interface facilitates changing the update interval (**R8**) and allows the simulation to be paused or restarted (**R9**). In addition, it provides a graphical interface for loading and saving the network state (**R10**).

2. Region selector: This service provides an abstract vision of the network structure and allows a way to select the different regions or populations which conform the network model.

3. Activity plot: The activity plot service shows the state of the activity for the selected regions/populations as it changes through the simulation.

4. Connectivity plot: The connectivity plot service shows the total number of connections for the selected regions/populations as it changes through the simulation.

5. Color editor: Allows the user to specify the colors and styles for the plotting services.

6. Manipulation of structural plasticity parameters: The user interfaces for η and ν are the primary means of steering the simulation for the parameter space exploration (**R6** and **R7**). This interface allows for the modification of the control signals, enabling the structural plasticity algorithm to take the system from its current state to a desired final state (see Figure 5.1, bottom center). Both steering interfaces are designed as separate standalone services that can be started within the *Control Panel*.

Figure 5.1 shows a snapshot of the ISV system. Please refer to the supplementary material of Nowke et al. [114] for a video which explains the detialed usage of these tools.

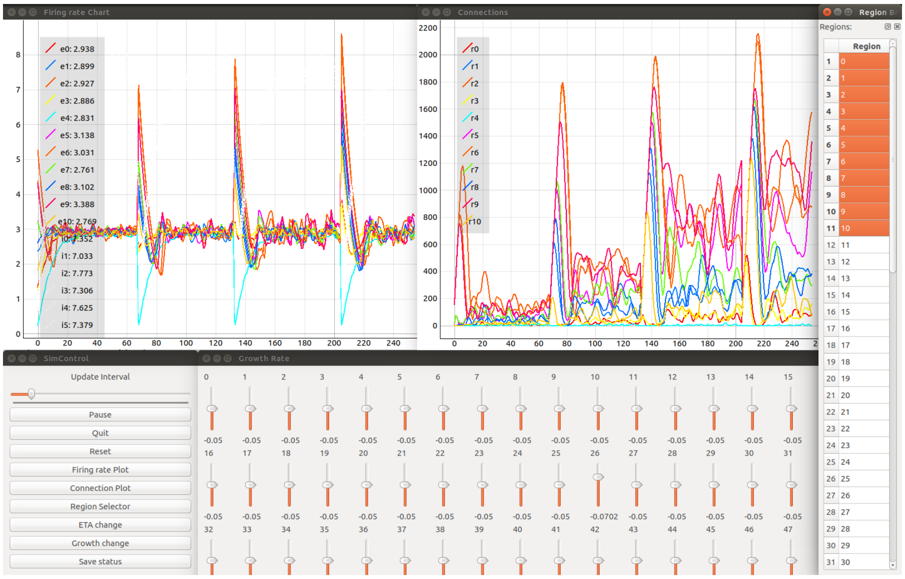


Figure 5.1: Firing rate in spikes/s of simulated brain regions (upper left) and total connections (upper right) are retrieved while a NEST simulation is performed. The time is measured in number of update intervals. The steering interfaces (*Control Panel* and growth rate manipulation; bottom left and center) allow interactive parameter space exploration which is synchronized with the current simulation. The growth rate (in Δ synaptic elements/ms) for each region can be controlled using the corresponding slider. The region selector (far right) provides the means to filter the brain regions of interest depicted in the plots. The legends provided in each plot denote the current selection from the region selector along with the color used to identify the corresponding curve. Specifically in the example shown, the labels *e0* - *e10* and *i0* - *i5* identify the average firing rate for excitatory and inhibitory populations in network regions 0-10 accordingly. Labels *r0* - *r10* identify total outgoing connections from network regions 0-10. Please refer to Section 5.4.2 for more details on the network model used in this example.

To re-use previously found connectivity patterns in neighboring points of the parameter space, we also implemented a save and load functionality (**R10**). The current values for η and ν are saved for each population as well as the connectivity update interval. All current connections between all neurons are also saved. These connections are defined by a source neuron, a target neuron and the synapse model which links them. Finally, the total number of connections for each population are exported to a file which can be used in the next phase of the simulation loop.

5.4 Application of the ISV framework on use cases

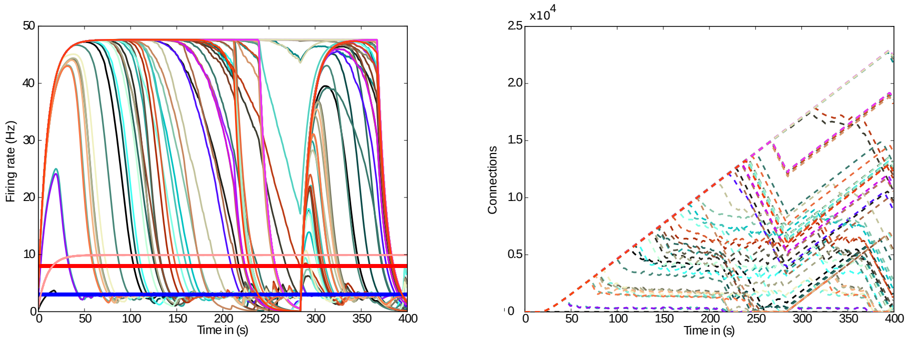


Figure 5.2: Previous method of visualizing simulations: visualization of the simulation as performed before the presented tool was developed. The figure shows the evolution of the average firing rate for each region (solid curves) and numbers of outgoing connections (dashed curves) from each region using structural plasticity in a non-interactive (static) experiment. Each color represents a different population. In this static approach, a large number of independent simulator runs are performed over a predetermined, non-interactive parameter space and then displayed with ad hoc scripts. Mapping the non-physiological solutions with saturated firing rates onto regions of the parameter spaces is highly non-trivial (compare approach with Figure 5.1).

In this section, I present the results obtained from two use cases in connectivity generation. For the first use case, the results of running structural plasticity simulations before the interactive visualization tool was developed were previously reported in Diaz-Pier et al. [37, Figure 5, section 3.3.1]. Figure 5.2 shows the equivalent output for the second use case, reflecting the previous visualization approach. Due to the large number of unlabeled curves, the inability to focus on data for particular populations and the lack of interactivity with the visualization, using this static approach makes it very difficult for the user to identify the evolution of connectivity in relation to parameter changes. Moreover, a new simulation run is required whenever any parameter needs to be changed. Even when some regions have easily reached the target activity of 3 spikes/s, for some set-ups it is extremely challenging to identify suitable trajectories that lead to stable solutions for all populations.

In this type of simulation, the system is constrained by connectivity data and desired activity levels obtained from experimental measurements. However, these constraints still allow the system to reach non-physiological states such as saturating at high firing rate values (see Figure 5.2). Moreover, the system may follow several trajectories to reach these implausible states, indicating

Parameter	**Value**
Capacitance of the membrane C_m	0.25 nF
Resting potential V_L	-65 mV
Threshold membrane potential V_thr	-50 mV
Reset membrane potential V_res	-65 mV
Refractory time τ_ref	2 ms
Growth rate excitatory synaptic elements	0.0001 elements/ms
Growth rate inhibitory synaptic elements	0.0004 elements/ms

Table 5.1: Network parameters for the first and second use cases.

that the system is under-constrained. On the other hand, there are many admissible trajectories which take the system to biologically plausible states. Biologically meaningful trajectories should be identified by heuristics, expert knowledge, and further experimental measurements gained through a deeper understanding of the parameter space to which the neural circuit is subject. At first glance, it is not clear how to explore the parameter space in these complex systems, as the large number of variables and long simulation times make it unfeasible to find stable populations through a brute force approach, and no heuristic is available to reduce the dimensionality. Without expert knowledge in a closed loop setup, admissible trajectories are fundamentally hard to find. In the following sections, I demonstrate the challenges of parameterizing network models and the potential of the ISV tool to address them. I have performed all experiments with NEST 2.10.0 [17] and its Python language bindings which are described in Eppler et al. [48], Zaytsev and Morrison [167]. The complete NEST scripts used in this work can be found in a GitHub repository which can be found in the supplementary material of Nowke et al. [114].

5.4.1 Two population model

In this use case, I create a model with two populations of point neurons, one excitatory and one inhibitory as shown in Figure 5.3a. The whole network contains 1000 leaky integrate-and-fire neurons with exponential shaped post-synaptic currents, of which 80% belong to the excitatory population and the rest to the inhibitory population. Parameters for the point neurons are listed in Table 5.1 on page 86. All neurons receive independent background excitatory Poisson noise at a rate of 10 kHz. At the beginning of the simulation, no connections between neurons are present. The system is allowed to create both excitatory and inhibitory connections (red and blue dashed arrows, respectively, in Figure 5.3a), using the structural plasticity framework in NEST.

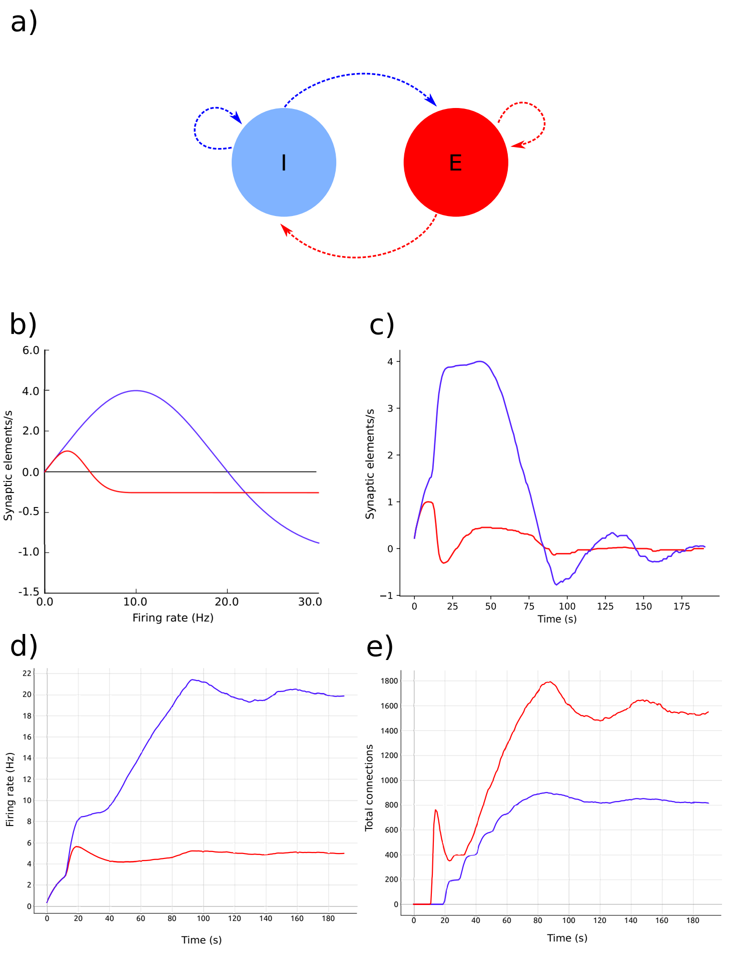


Figure 5.3: Evolution of firing rate and connectivity for the two population example. a) Abstract view of the model consisting of two populations, one excitatory (red) and one inhibitory (blue) with respectively excitatory connections (red arrows) and inhibitory connections (blue arrows), both controlled by structural plasticity; b) Gaussian growth curves mapping current firing rate to growth rates (see Figure 2.2); c) growth rate dynamics; d) evolution of the firing rate; and e) evolution of the total number of connections during the simulation. Colors in (b)–(e) are as in (a).

The weights for the created synapses are 1 and -1 respectively. The evolution of the firing rate (Figure 5.3c) and the growth of connections (Figure 5.3c) is regulated by two homeostatic rules defined by Gaussian curves, as shown in Figure 5.3d. The target average activity of the inhibitory population is set to 20 Hz while the target average activity in the excitatory population is set to 5 Hz. Figure 5.3c shows the evolution of the growth rate for excitatory synaptic elements in both populations during a simulation. These dynamics originate from the fixed firing rate curves shown in Figure 5.3b. The structural plasticity algorithm uses that relation at every simulation step to decide how many connections to create or delete.

The evolution of the connectivity generation can be guided by modifying the growth rate and shape of the Gaussian curve linked to each type of connection. Figure 5.3d and Figure 5.3e show an example of this process. In this use case, an interesting feature to observe using the visualization and steering tool is the path to the solution. With the configurations used here, one can see how allowing faster growth of inhibition triggers an overshoot in the generation of excitatory connection to compensate. As a result, a rewiring of the system is obtained. These paths to the solution can be linked to onsets of critical periods in learning and healing or by external stimulation [69]. By regulating the speed of the creation of connections in the system, scientists can explore different paths to solution where the relationship between excitation and inhibition changes through time.

Figure 5.4 shows the evolution of growth rate (synaptic elements/s), firing rate (Hz) and connectivity (total number of connections) for six examples of the multiple trajectories and connectivity configurations that the network can show. All examples start with an initial growth rate of 0.0001 synaptic elements/ms. Figure 5.4a shows a smooth growth similar to Figure 5.3, but where the control signals have been modified to reduce the overshoot in the inhibitory population. That is done by reducing the initial growth rate to 0.00005 at iteration 8 (mark a.1). Figure 5.4b shows an example of a simulation where the control signals for growth start with aggressive growth values, producing a constant oscillatory behavior. That is achieved by changing the growth rate from 0.0001 to 0.0010 at iteration 38 (mark b.1) and then to 0.0030 at iteration 80 (mark b.2). Following these signals, the connectivity update interval is increased to 500 ms (from the standard length of 100 ms), which produces a big oscillation, triggering a rewiring of the network (mark b.3). Finally, growth is reduced to a slower pace, which helps the system settle at a stable state. This reduction is achieved by setting the growth rate to 0.00005 at update 161 (mark b.4). The final connectivity is very similar to the one reached in Figure 5.3. This example shows a different trajectory which reaches the same final state.

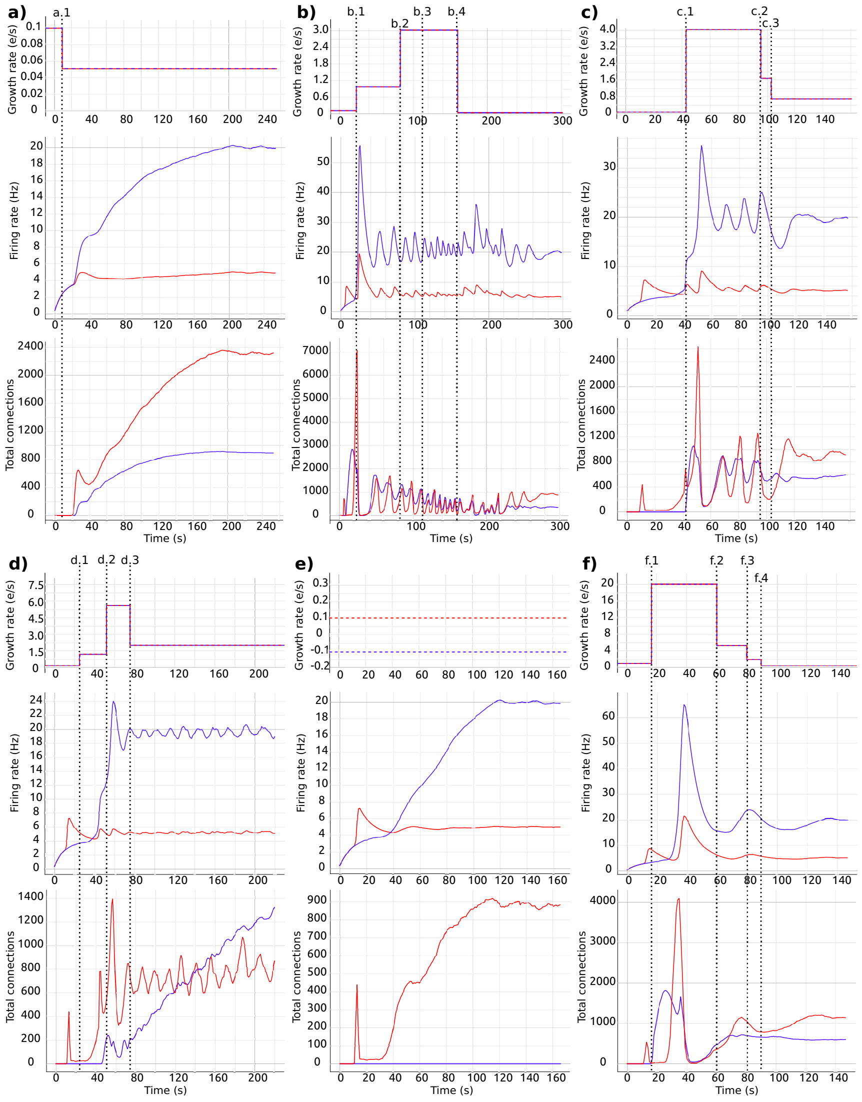


Figure 5.4: Evolution of growth rate (top), firing rate (middle) and outgoing connections (bottom) for six different trajectories (a-f) in the two population model use case, excitatory (red) and inhibitory (blue). Vertical dashed lines correspond to manual changes using the graphic interface to the growth rate (top curves) or update interval (at b.3) control variables. All other simulation parameters are held constant for all runs, including initial growth rate. Please see the main text for a discussion of the features of each set of trajectories.

Figure 5.4c illustrates very fast initial growth by changing the growth to 0.004 at iteration 46 (mark c.1). Then, a sharp reduction in growth when the system oscillates near the target firing rate. The growth rate is changed to 0.0018 at iteration 98 and further down to 0.0007 at iteration 103 (marks c.2 and c.3 accordingly). Figure 5.4d shows a case which seems stable in terms of activity, but is unstable in terms of connectivity, as it exhibits a constant race between excitation and inhibition in the connectivity to maintain the target activity. The growth rate is set to 0.001 at iteration 24 (mark d.1), to 0.0056 at iteration 52 (mark d.2) and to 0.0020 at iteration 78 (mark d.3). Figure 5.4e shows a trajectory which is not biologically meaningful. This network has been built only from excitatory connections by modulating the growth of connections very carefully around the target activity. Here, we have defined a growth curve that does not allow the creation of inhibitory connections unless the activity is above the desired firing rate. Finally, in Figure 5.4f, we see a trajectory which is not admissible (not biologically meaningful) because the network is taken to an artificially high firing rates before it settles back to its target. At iteration 17, the growth rate is set to 0.002 (mark f.1) and then slowly reduced to 0.00056 at iteration 60, to 0.0002 at iteration 80 and finally to 0.00005 at iteration 90 (marks f.2, f.3 and f.4 accordingly). This graph shows how a network can traverse biologically inadmissible trajectories and still reach the target activity.

These results show that several instantiations of the same system using different dynamics lead to the same target activity but different connectivity patterns. Visualization and steering is fundamental for producing, observing, studying and cataloging these behaviors in the network. See Bahuguna et al. [9] for an example of the same phenomenon exhibited in a more complex network. Thus using target activity as a tuning parameter without this kind of exploration leads to selecting one of these network connectivity states arbitrarily. The resulting model may not be representative of the kinds of networks that produce this activity, or of the target system to be modeled. In other words, the target activity does not uniquely identify a network, or even a contiguous volume of parameter space, but is the property of a distribution of distinct networks distinguished by parameters that are not the direct targets of research — this class of inverse problem is degenerate. The network structure may be critically path-dependent, dependent upon parameters which are stochastic sequences (external control variables) or even dependent upon numerically unstable parameter functions. Simple networks such as the one shown in this example are frequently used in computational neuroscience but rarely with consideration to the careful characterization of the parameter spaces. Thus, in the absence of analytical methods to identify alternative solutions in the parameter space, steered visualization is a highly effective method for

Parameter	**Excitatory Neurons**	**Inhibitory neurons**
Number of neurons N_{r}	160	40
Capacitance of the membrane C_{m}	0.5 nF	0.2 nF
Membrane leak conductance g_{m}	25 ns	20 ns
Resting potential V_{L}	−70 mV	−70 mV
Threshold membrane potential V_{thr}	−50 mV	−50 mV
Reset membrane potential V_{res}	−55 mV	−55 mV
Refractory time τ_{ref}	2 ms	1 ms

Table 5.2: Network parameters taken from Deco et al. [35] for each region.

producing, observing, comparing and cataloging network configurations.

5.4.2 Whole brain simulation

This use case is inspired by the previous study of Deco et al. [36]. The experiment consists of a whole brain simulation using 68 interconnected brain regions, each of which represented by a spiking network containing 200 conductance-based leaky integrate-and-fire neurons, as illustrated in Figure 5.5a. The original work by Deco et al. uses a Dynamic Mean Field Model (DMFM) originally developed in Wong and Wang [163].
The coupled non-linear stochastic equations of the DMFM describe the behavior of mean-field neuronal regions and their influence on each other:

$$\begin{aligned}\dot{\mathbf{s}} &= \mathbf{s}/\tau_s + (1-\mathbf{s})\gamma H(\mathbf{x}) + \sigma\boldsymbol{\nu}(t)\\ H(\mathbf{x}) &= (a\mathbf{x}-b)/\Big(1-\exp\big(-d(a\mathbf{x}-b)\big)\Big)\\ \mathbf{x} &= wJ_N\mathbf{s} + GJ_N\mathbf{C}\mathbf{s} + I_0\end{aligned} \tag{5.1}$$

where H represents the population firing rate function; $\mathbf{s}$ is the vector representing the average gating variable for each region; a, b, d, and σ are scaling parameters; γ and τ_s are kinetic parameters; $\boldsymbol{\nu}$ is the stochastic input vector; w is the local excitatory recurrence; J_N is the synaptic coupling; G is the general coupling factor; $\mathbf{C}$ is the connectivity matrix; I_o is the effective external current; and $\mathbf{x}$ is the state variable vector for the regions. This model is applied in Deco et al. [35] to describe a system dominated at the measured time frame by NMDA gating, while AMPA and GABA gating are neglected as 'fast' variables. For a complete description and analysis of the model, see Wong and Wang [163] and Deco et al. [36].

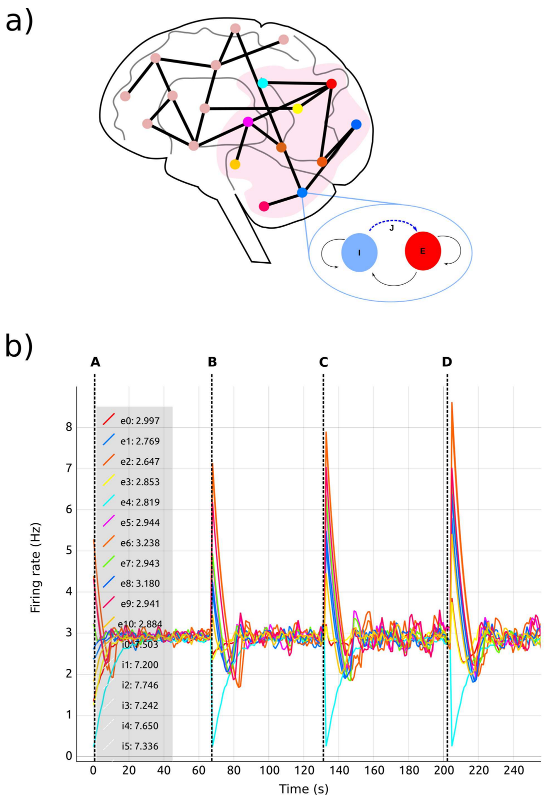


Figure 5.5: Use case 1 inspired by Deco et al. [36] whole brain model. a) Abstract representation of the whole brain model including 68 regions. A subset of the regions is selected (pink area). The zoom-in view of one of the regions shows the abstract model of each region, consisting of two populations, one excitatory (red) and one inhibitory (blue). Inhibitory connections to excitatory neurons in the same region (blue dashed arrow labeled *J*) are subject to structural plasticity. b) *Activity Plot* of selected regions $(0-10)$ as a function of biological time. Regions are numbered from 0-67. Tags *eX* and *iX* identify curves for excitatory and inhibitory populations in the *Xth* region. A legend (upper left) indicates the current selection. The number following the colon after the tag is the region's firing rate during the last simulation step. Vertical dashed lines separate sections of the simulation with differing values of the global connectivity coupling (see Section 5.4) , $G =$ A) 0.5; B) 1.0; C) 1.5; D) 2.0. The vertical dashed lines are superimposed on this plot and are not part of the *Activity Plot* service. Increases to the global coupling parameter lead to an increase in the strength of the connections between regions. The firing rate spikes initially as a response to this change; in response, structural plasticity modifies connectivity according to the homeostatic rules until the firing rate stabilizes again closer to the target firing rate.

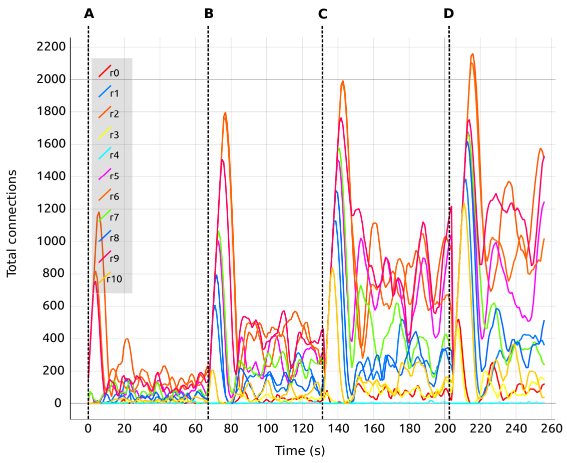


Figure 5.6: Total number of connections for selected regions $(0-10)$ as a function of biological time. Colors are synchronized between this plot and the *Color Editor*. Vertical dashed lines separate sections of the simulation with differing values of the global connectivity coupling (see Section 5.4), $G =$ A) 0.5; B) 1.0; C) 1.5; D) 2.0. Regions are coupled and numbered from 0-67. Tag *rX* identifies the curve for the total number of connections corresponding to the *Xth* region.

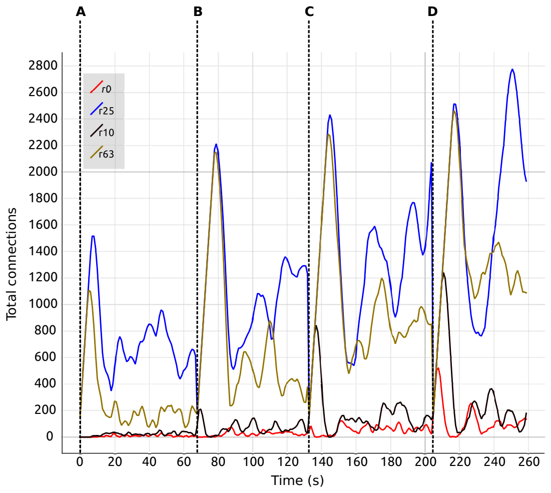
Here, I apply a mapping from the DMFM to a network of point neurons. In this simulation, each region contains two populations, one excitatory (80% of the total neurons in the region) and one inhibitory (20%). In this case, neurons in NEST do not represent biological neurons but processing units whose mathematical description at population level is equivalent to the elements which comprise the DMFM. The initial parameters used to set up the network are taken from Deco et al. [35] and detailed in Table 5.2 on page 91. For a complete explanation of the model and its motivation, see Deco et al. [35] and Wong and Wang [163].

Recurrent excitatory connections have a strength of 1.4 pA, while recurrent inhibitory connections have a weight of -1.0 pA. Each neuron per region initially receives 160 excitatory connections from the local excitatory population. Only inhibitory connections are created during the simulation (blue dashed arrow tagged J in the region zoom in of Figure 5.5a) since we are only interested in substituting the feedback inhibition control algorithm described by Deco et al. [36]. The inter-regional connectivity (black lines between regions in Figure 5.5a) is specified from structural data obtained by DTI, which results in a connectivity matrix C, but further regulated by the general coupling parameter G, a multiplicative factor. This enables the linear modification of the strength of the connections without altering the ratio of connectivity among regions. Thus, the total weight of the connections between regions is equal to $G \cdot C$ pA. Each connection between regions is made between a single representative neuron in each excitatory population. Connections between regions are only excitatory. Additionally, all neurons receive independent background input from a Poisson generator producing spike trains with a rate of 11.9 kHz.

I followed the procedure described by Deco et al. [36] for the generation of synaptic activity, substituting the feedback inhibition control used in that paper with the interactive exploration method. The strength of the background input was tuned to achieve a firing rate of 3 spikes/s for the excitatory population and 8 spikes/s for the inhibitory population when regions were isolated (without inter-region connections). In Deco et al. [36], an iterative tuning strategy was used to determine the intra-region inhibition for the DMFMs required to produce an activity profile consistent with experimental observations. The key insight inspiring our approach is that finding the intra-region inhibition can be mapped on to determining the number of inhibitory connections required to produce the same activity pattern in a multi-area spiking neuronal network. Finding the right amount of inhibition per region which satisfies all the dependencies is still a hard multi-objective optimization problem, especially if the space cannot be interactively explored. This is demonstrated in Figure 5.2, which shows the result of simulating one static parameter setup for the connectivity generation programmatically.

In this setup, the tool was used with different values for the inter-region global coupling factor G. A complete view of the visualization and steering tool for this use case is shown in Figure 5.1. By using the tool, we detected that as G grows, it becomes more difficult to bring all regions to the desired activity state, and the standard deviation of the average firing rate increases as well. G has this impact because any change in one region due to G has a strong impact on all other regions dynamically reacting to the change in G. This effect is visible in Figure 5.5 and Figure 5.6, where the time it takes for all regions to stabilize increases as the value of G grows.

We are also able to detect which regions are more crucial for stability, since they have a higher inter-connectivity to other regions. Figure 5.7 shows a comparison of the evolution of the firing rate and outgoing connections of four regions. Each peak shows an increment in the global coupling value G by 0.5, starting from a base value of 0.5. Regions 25 and 63 show large oscillations due to their high connectivity with multiple other regions. Conversely, regions 0 and 10 rapidly reach a stable state even for high values of G. This capacity for detailed inspection allows the researcher to verify that all regions reach the desired average activity while the simulation is running, and thus drastically decreases turn-around times to research this behavior.



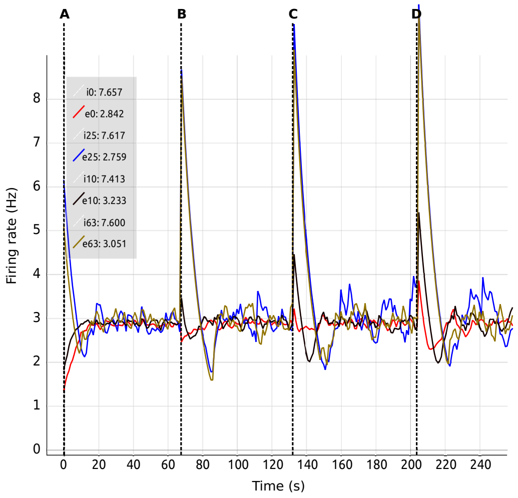


Figure 5.7: Number of connections (top) and firing rate (bottom) shown in comparison of four regions $(0, 10, 25, 63)$. Vertical dashed lines separate sections of the simulation with differing values of the global connectivity coupling, $G =$ A) 0.5; B) 1.0; C) 1.5; D) 2.0. Regions are numbered from 0-67. Tags *eX* and *iX* identify curves for excitatory and inhibitory populations in the *Xth* region. The number at the side of the tags denotes the current value of the average firing rate for each region.

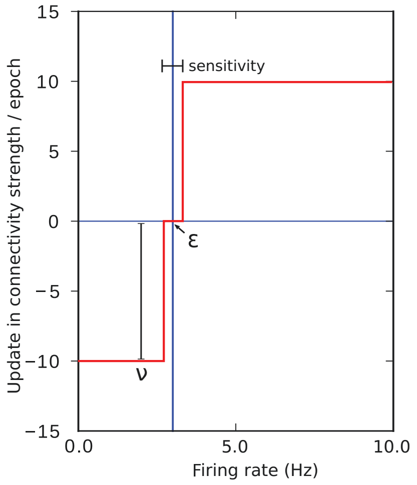


Figure 5.8: Connectivity update rule employed in the algorithms proposed in Deco et al. [36] and Schirner et al. [133], where a fixed value is added or subtracted iteratively to the inner inhibitory connectivity until convergence to the desired firing rate is achieved in each simulated region.

The search algorithms proposed in Deco et al. [36] and Schirner et al. [133] are based on an update pattern which (in the same terms as the algorithms used in this work) can be described by a fixed step update around the target activity, as shown in Figure 5.8. The effectiveness of a fixed search approach in the connectivity parameter space depends mainly on two factors. First, the size of the correction step. If the step is too small, it will take too long to reach the target activity if the initial conditions are not close to the solution. If the step is too large, the system will oscillate because the corrections are too coarse. Second, the effectiveness is dependent on the accuracy. The speed to find a solution is inversely proportional to the desired accuracy. The correction step should also be smaller than the accuracy, otherwise the system may oscillate indefinitely around the final target state without ever reaching a state with the desired accuracy. In summary, the ability of the search algorithm to find a solution depends on the initial conditions, the size of the update step and the desired accuracy. The approach I present here allows the size of the update step and the speed with which changes take place to be adapted during simulation. This solves the problem of the dependency between step and accuracy and also allows the system to potentially find a solution from a broader range of initial conditions due to the capacity to increase the resolution of the search as the target state is approached.

In addition to the advantages in speed and use of computational resources which our expert-steered approach confers over brute force parameter search (and which may, in fact, be computationally intractable), the process of steering allows the researcher more insight into the system. Whereas in the first use case, the primary finding was that multiple connectivity configurations can result in the same activity profile, in the second use case I am able to identify which regions are most critical for the overall network stability, as illustrated in Figure 5.7. Thus, interactive visualization can support the researcher in sensitivity analysis, which is essential for understanding the main driving parameters of the model and for making better inferences about the relations between parameters and function. As with the multiple configurations observed in the first use case, it is rare to encounter a network modeling study in computational neuroscience where a sensitivity analysis has been carried out (but see Bos et al. [18] for a counter example.)

5.4.3 Usage of the tool

In this section, I summarize the main steps required to use the tool to take the system from its initial state to a final connectivity setup where the target mean activity values are achieved. In the following, I make reference to the requirements listed in Section 5.3. The first step during the simulation steering is to determine which regions have one or more of the following characteristics (**R2-R5**):

- the electrical activity is far from the target activity, and there is no tendency of the system to correct for this error (or the correction is too slow);
- the electrical activity oscillates around the target activity and the oscillations are of equal or higher amplitude in each cycle;
- or the number of connections does not converge even though electrical activity is around the target activity.

This is achieved using the visualization tool by observing the firing rate and connectivity plots. Figure 5.5 shows the evolution of the firing rate for the first ten regions of the brain model. Figure 5.6 shows the changes in connectivity which are guided by the homeostatic growth rules defined for the structural plasticity algorithm. Each curve in the plot is uniquely identified by color and linked to a population or region, thus enabling the assessment of the three above listed characteristics. Reaching the targeted stable state is indicated when all firing rate curves converge to the target activities while the connection curves flatten to horizontal lines. This allows the user to simply and effectively identify which regions deviate from the target state and to correct the structural parameters according to the following criteria:

- If the actual electrical activity is far away from the target activity, the growth rate ν for that region should be increased (**R6**).
- If the actual electrical activity oscillates around the target activity, the growth rate ν for that region should be decreased in small increments and the value of η should be reduced to decrease the rate of change in the number of created and deleted synaptic elements around the target point ε (**R6-R7**).
- If the number of connections does not converge, highly interconnected regions should be identified and the growth rate ν should be modified down in all of them (**R7**). In this case, the update interval can also be modified to a smaller value to have a faster response of the control changes in the connectivity (**R8**). A shorter update interval allows better and smoother control, but impacts the performance of the simulation.

The resulting network state can be saved and used later as a starting point for other parameter combinations, thereby minimizing the need for further computations using similar values of the global coupling term (**R9-R10**).

5.4.4 Implementing further use cases

Using an event-driven architecture, the ISV framework provides a convenient way for domain scientists to extend the tool to their needs. This tool can be used with any neuron and any synapse model in NEST, except for gap junctions. By using the scripts provided in the supplementary material of Nowke et al. [114] as templates, the user can easily change the neuron and synapse model to explore the impact of these variations. An instrumentation manual which specifies the steps required to integrate the tool with other network models implemented in NEST can be found as part of the supplementary material. The instrumentation manual provides instructions based on examples for NEST, but the tool can be adapted to other simulators providing a Python interface by replacing the corresponding functionality. However, if the simulator does not provide an interface to Python, instrumentation will require substantial development effort by the user. Table (5.3) provides estimates for the complexity of adapting the *nett* messaging library to different use cases. The complete tool and the underlying messaging framework is open source.

Challenge	*Solution*	*Complexity*
Change network topology	Change number of populations	simple
Increase the size of the network	Increase the number of neurons in the simulation script	simple
Retrieve additional parameter	Create new event definition	medium
Add a new model parameter	Create new event definition	medium
Connect different simulators with Python interface	Create new event definition for the specific simulation values	medium-hard
No Python / C++ environment	None	hard

Table 5.3: Estimation on the complexity to adapt the *nett* messaging framework to different steering and visualization use cases.

5.4.5 Simulating on a supercomputer

To leverage the power of supercomputers to reduce turn-around times for parameter space exploration, the simulation scripts can be adapted to use MPI. In this section, I show an example of adapting the whole brain simulation use case described in Section 5.4.2 to supercomputers. To ensure that each process is in sync with all steering commands, one process (rank 0) serves as master. Only this master process establishes a connection to the visualization front-ends and processes their steering events. Then parameter synchronization is conducted via synchronization barriers with the remaining compute nodes. The master process is responsible for gathering the electrical activities and total connections from all other compute nodes to finally send these to the visualization front-ends.

After all the simulations had been parallelized, the tool was adapted to cope with the supercomputing environment. A challenge of the current usage conditions of most supercomputing environments is their batch-mode operation where users submit jobs which are granted compute time after a possibly long delay; interactive supercomputing is still a work in progress as outlined in Lippert and Orth [95]. Since our tool relies on a network connection to NEST, the IP-address of the compute-node running the simulator is unknown *a priori*. To circumvent this issue, the supercomputer's global file system was used: when the simulation is granted compute time, the node's IP-address is obtained and written to disk. Subsequently, all visualization services use this configuration file and connect to the given address. However, one limitation of this approach is the need to start the simulation first.

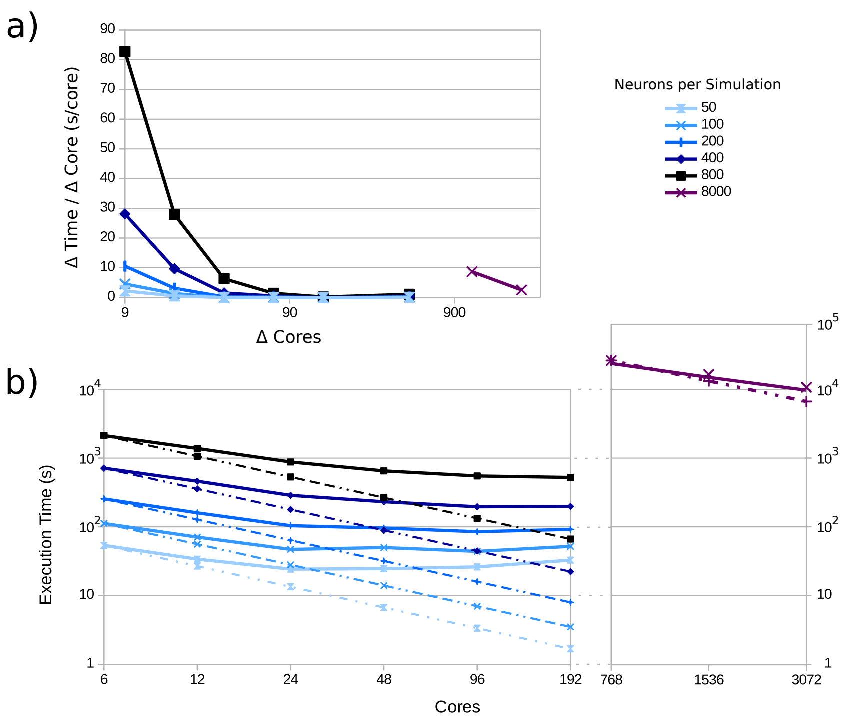


Figure 5.9: Execution time as a function of the number of compute nodes for varying numbers of neurons per population: 50, 100, 200, 400, 800, and 8,000; curves shaded from light to dark. Dotted lines indicate ideal scaling, while solid lines represent experimental results. a) The change in time consumed per core added. Each point is the difference in the execution time divided by the difference in cores for consecutive points in (b); points are placed at the midpoint between the source measurements. b) The execution time for each simulation as the number of cores are varied; for the simulation with 8,000 neurons per populations, measurements were made only for 768–3072 cores, since fewer cores leads to excessive time demands. The simulated biological time was 5s using an update interval of 100 ms.

Since the visualization tools are independent of the network topology and size, the scaling impact of the network's performance can be measured while neglecting the communication overhead. To simulate a larger number of populations with a larger number of neurons, it is crucial to use supercomputers. To this end, I deployed the tool to the JURECA supercomputer at the Jülich Super Computing Centre. JURECA has 260 compute nodes with Intel Xeon E5 – 2680 v3

Haswell CPUs with 2×12 cores per CPU, 128 GB of RAM per node and runs CentOS 7. To assess the speed-up obtained using this machine, we used the third use case's setup and measured the execution times for 50 updates of connectivity in the network, with an update interval of 100 ms. Using a full node on JURECA, I am able to obtain a 2.94-fold speed-up compared to the workstation setup, which uses 8 Intel Core $i7-4710MQ$ CPUs @ 2.50 GHz and 16 GB of RAM running on Ubuntu 16.10.
Figure 5.9 shows a strong scaling test for different numbers of neurons per population. Simulation scalability increases with the number of neurons per population — particularly for $8,000$ neurons per population (a total of $544,000$ neurons in the network). This is due to the network size and spike distribution overhead; larger networks benefit more from the larger number of compute-nodes and overcomes the inter-process communication and intra-process spike distribution overhead up to the point that the global number of spikes dominates performance (for a current discussion, see Jordan et al. 77). In addition, the number of synapses increases quadratically with the number of neurons per population, which highly impacts the scalability of the simulation.
On the other hand, visualization scalability is dominated by the data gathering step at every update interval. For the case of large networks like the $8,000$ neuron network, the impact of the data gathering step can be reduced by gathering information from only a portion of the network. A well selected statistical sample would provide enough information about the ensemble behavior of the populations while benefiting performance. The current paradigm for the tool funnels data from a large number of compute backends to a single frontend visualizer. In order to scale with increasing numbers of backend nodes for massive supercomputing, a more complex data flow and analysis framework will be needed, such as a multi-node reduction stage to reduce the impedance between the backends and frontend, as well as reducing the load on the fronted. A generalized software framework for such infrastructure to couple visualization with supercomputing at scale is, to my knowledge, currently not available, but considered in the community for future development.

5.5 Discussion

In this chapter, I have introduced a visualization and steering tool for the interactive analysis of connectivity generation in NEST. To show its applicability, I have presented two use cases where the tool was used to visualize and steer populations of point spiking neurons to reach a desired target activity level. My results indicate that by interactively exploring the parameter space and possible trajectories, scientists can gain a better understanding of the system and concentrate on

regions of biological interest, as compared to a blind brute force exploration. The improvement over brute force exploration is because the effects of changes in specific parameters in the network can be visualized and states outside the admissible regions can be identified and excluded from further simulation. This improvement leads to a reduction in computational resources and an educated definition of interesting parameters, states and trajectories.
In this chapter I have used firing rate calculations produced internally in NEST to guide the generation of connectivity. This method for computing the mean firing rate can impact the performance of the control system since controllability depends on the delay between measuring an observable and producing a response. However, the tool is independent of this calculation and other techniques, such as spike train binning, can be used instead to increase the controllability. Calculation of firing rate on streams of spike trains might become computationally intensive with increasing network size. Future implementation of other techniques to increase the data gathering speed will lower delays and allow other spike processing techniques to be efficiently implemented as alternatives to the convolution approach.
I selected the use cases in this chapter for their differing degrees of complexity in terms of connectivity and network definition. In the simple use case, different connectivity configurations lead to the same activity profiles even when some of the trajectories are biologically inadmissible. With the approach used in this work, the user can concentrate on exploring only those configurations which are of interest in answering the scientific question posed. In Figure 5.4, I show different trajectories produced using structural plasticity following homeostatic rules to fit the system to a firing rate profile. Even the non-biological parameters of the optimizing algorithm itself have an impact on the final configuration of the network. For example, if one performs a gradient descent to optimize the activity profile of a network, the results will be sensitive to any arbitrary choice of initial states of the populations and connectivity. With the ISV tool I was able to characterize the distribution of representative models and results.
In the second use case, the ISV tool also enables a sensitivity analysis of the system by visualizing the effect that changes in the connectivity have on the dynamics of the full system. Thus, the user can draw better conclusions about the relationships between the controllable parameters, in this case connectivity, and the observables of the system, in this case the firing rate of each population. The relative sensitivity of the system to the biologically relevant parameters (connectivity) and the non biological parameters of the optimization algorithm can be observed. The ISV tool can provide more insight into how different types of synapses are created or modified in the neural circuit to give rise to different features in the dynamics of the system.

As I have discussed in this chapter, a brute force exploration of the parameters of a network can easily become a computationally intractable problem. Choosing a single random configuration or even only a small sample of configurations from the whole space without a proper characterization of their distribution is unlikely to lead to a statistically valid distribution of the results. Interactive visualization is a way to move toward statistically validated conclusions as it allows an assessment of the essential features of the system, ultimately leading to automated sampling.
While the resulting connectivity patterns are not necessarily unique, the approach I discuss here enables exploration and assessment of these solutions and their paths. The main contribution of this approach is the use of interactive visualization and parameter control techniques. These techniques allow the system to be controlled and stabilized within a physiological configuration space by an expert. When increasing the number of neurons and populations, the number of parameters to tune increases, resulting in an ever harder-to-reach stable state. Thus, interactive visualization becomes even more important. The knowledge gained through interactive exploration can lead to the development of automated tools assisting in the parameter space exploration.
Using this approach, the turn-around times of exploring different connectivity configurations can be reduced in comparison to simulating all possible parameter configurations and assess reasonable configurations in a later phase. The speed-up achieved by this exploration is mainly due to four factors. First, it is not necessary to simulate the system for long times iteratively; instead, the modifications are performed on demand. Second, partial solutions can be reused for different global parameter combinations, resulting in the reduction of total computational costs. Third, the user can visualize the behavior of the system's observables with respect to individual parameters, allowing to isolate regions of interest and form a better understanding. Finally, we can study the transition points in the activity of the networks, which are produced by the underlying connectivity variations, and interact with the tuning algorithms by visualizing their impact.
As stated before, the connectivity solutions and paths to solutions for the presented use cases are not unique, rendering a knowledgeable exploration process crucial. Thus, the interactive analysis process can accomplish the following:

1. Form an understanding of the implication of different parameter setups for each network model.
2. Validate the models.
3. Define biologically meaningful populations of interest for the simulation.

4. Derive measures for the automatic or semi-automatic assessment of the models' behavior leading to automated tools guiding the exploration process.

While the generation of connectivity based on empirical constraints for the dynamic system or experimental data inherently leads to non-unique solutions and especially solutions which are physiologically implausible, the ability to identify and explore subsets of the solution space is valuable to form an understanding of the dynamic nature of these systems.
In this work, I have formalized the effects of dynamic connectivity of a network in terms of control theory. I take into account that the network starts at an initial state and is taken to a final state through the introduction of control signals which alter the connectivity of the network. In this case, control of the synapse creation and deletion is induced by the structural plasticity algorithm. The eigenvalues of the Liouvillian of the network are thus modified with these signals through the evolution of the simulation and the state of the neurons in the network is changed. Visualization shows the immediate effects of the control signals in the system. The results shown in Section 5.4.1 exemplify how even a simple network can traverse different admissible trajectories (Figure 5.4 a-e) using different elements from the set of all possible controls. I show how the unconstrained system can traverse an inadmissible trajectory (Figure 5.4 f) or end in states outside of the admissible set (Figure 5.4 d). One can also seen how inadmissible control signals are still able to give rise to admissible trajectories and final states (Figure 5.4 e).
The ISV tool was adapted to scale with supercomputers allowing larger networks to be simulated and finer simulation stepping to be used, thus achieving more accurate results. This way, researchers can explore the manifold solutions and paths of connectivity satisfying average activity targets in a variety of neural network models. The tool gathers data from the simulation at specific intervals, which impacts the performance as the networks become larger. Continuously streaming data from of the simulation by using, for example, MUSIC [38] or the NEST I/O backends can reduce this bottleneck and allow greater flexibility in the network size.
In summary, the tool presented in this chapter provides the means to visualize and steer connectivity generation of a running NEST simulation to stabilize complex non-linear systems. The applied concepts of the tool are generalizable and extensible to other types of systems with similarly large degrees of freedom. Adapting and exploring further model parameters, e.g., synaptic weights and delays, background input frequency, and variation in weights of spike-timing-dependent plasticity synapses is possible.
In the future, other techniques to track already explored parameter spaces, to develop semi-

automatic systems to guide researchers in tracking manifold solution spaces and to extend the tool to support further use cases could be explored. Currently, the saved state refers only to the connectivity and last values of all variables at the time of saving. A proper visualization which shows parameter changes required for reproducing every trajectory would be very helpful to further understand the sensitivity of the network to specific connectivity changes. For the moment, the loading features are limited and the subject of future work. Machine learning algorithms can also be coupled with the interactive exploration for various network variables beyond connectivity (see Chapter 7). As an external observer, learning algorithms could detect oscillations and other troubling behavior in the network and correct this behavior using the steering controllers. Other target control measures such as power-spectrum shape and inter-population correlations may be interesting as complex control variables in the context of machine learning. The modularity of the software, primarily derived from applying an event driven design, allows for such additions in a non-intrusive manner.

Linking the time axes of the activity and connection plots to allow for coordinated zooming is currently not supported but would be a useful extension to the analysis workflow. A visualization of changes in the network's eigenvalues as connectivity evolves is also subject to future work. The creation of additional plots for further variables is simple and can be achieved by adapting the scripts used in the presented use cases (see the supplementary material of Nowke et al. [114]). Connecting another visualization application to the NEST simulator is in principle feasible but requires adapting the visualizer to our communication protocol.

In Nowke et al. [114] we argue that it is crucial to explore the distribution of paths to solutions instead of focusing on just *a* possible solution satisfying a set of constraints. To develop this understanding, interactive exploration of dynamic systems is a key tool for developing mathematical intuition, and thus for deriving mathematically robust descriptions. These descriptions are then amenable to further automated investigation of characteristic solution ensembles.

While in Chapter 4 I showed a static view of structural plasticity simulations, the tools developed and described in this chapter allow for a better analysis and understanding of the changes in the structure and dynamics of the simulated networks. The ability to change the structure of a spiking neural network model during simulation is an additional tool provided to the end user to observe and explore the sensitivity to input/output changes in different populations of the network.

6 Clinical applications

In the previous chapters I have detailed the implementation of model structural plasticity in NEST together with a set of visualization and analysis tools useful to understand the impact that changes in the connectivity of spiking neural networks have in its function. In this chapter I present an application of structural plasticity modeling to a concrete scientific question in the realm of clinical neuroscience. Here, the application of Coordinated Reset Therapy (CRT) is simulated on a model of the Sub Thalamic Nucleus (STN) and the Globus Palidus externus (GPe). These two regions of the brain, are involved in unhealthy synchronization in Parkinson's disease. In order to simulate long lasting effects of CR therapy, this model considers both synaptic and structural plasticity. Synaptic plasticity is modeled using Spike-Timing-Dependent Plasticity (STDP). This work shows how the model can be used to simulate neuorbiologically meaningful phenomenon and supports its base as a connectivity parameter search exploration technique with direct applications to neuroscience.

In [103] we propose a model to study the impact of structural plasticity in this kind of therapy. We propose to use a combination of synaptic and structural plasticity in a model of the Sub Thalamic Nucleus (STN) and the Globus Palidus externus (GPe) in order to study the impact of these plasticity effects in relation to CRT. It has been observed that the effectiveness of CRT shows a cumulative effect, allowing subsequent applications of this procedure to have stronger and faster therapeutic effects. As clinical data is hard to obtain, a simulation strategy may enable to predict dosage-dependent phenomena relevant for clinical studies. Several simulations of CRT have been proposed and performed in the past [146, 145, 144]. However, the long lasting effects of CRT have not yet been visible in such simulations. With this work we offer the scientific community a modeling and simulation framework to study and hopefully inform the way iterative CRT could maximize its effects.

6.1 Introduction to structural plasticity in clinical applications

Structural plasticity mechanisms related to healing have been studied for long time in clinical research. One large area of research has focused on explaining the mechanisms behind rewiring which occurs after injury, including sensory deprivation and stroke. For example, it has been observed that structural changes, similar to those observed during brain development, are triggered soon after a stroke takes place. This is seen by an increase in growth-associated proteins in the peri-infarct areas during the first two weeks after the event, which is then accompanied by synaptogenesis [141]. This seems to be a natural response of the brain trying to compensate for the disruptions caused by the injury.

There are several factors which play an important role in the responses of the brain to injury and disease but the basic mechanism which regulates this adaptation, recovery and compensation is learning. This topic is discussed in [92], where the authors explore the role of learning in the rewiring of the brain but also how injury and disease affect the way learning takes place in the brain. The authors highlight the need of therapeutic measures to consider how learning is affected and changes by disease but also by the natural reactions of the brain and the therapy itself.

In [153], the authors experimentally observed the structural changes that pyramidal neurons in the cortex suffered during deprivation of sensory stimuli. They observed that, even when the general branching of the dendritic tree remained mostly unaltered, there was a high level of spine sprouting and retraction. They determined that about 50% of the spines only had a lifespan of a few days or less and concluded that sensory experience is linked to a high synapse turnover.

Neural synchronization, the simultaneous activation or large number of neurons in rhythmic patterns, has been identified as a common feature in different mental diseases such as Parkinsons' and tinitus. Coordinated Reset Therapy (CRT) is an alternative to reduce the synchronization of neural activation and aids in the neural network 'rewiring' which leads to reduced symptoms of these diseases [144, 145]. CRT is achieved by inserting stimulation electrodes in the brain region of interest and generating a pulsed signal. The characteristics such as time of exposure, frequency and alternancy in the activation - deactivation among electrodes can be adjusted to meet an specific stimulation protocol with different therapeutic effects.

It has been shown that CRT enhances the quality of life of patients with a suitable profile by reducing the strength of symptoms. In the treatment of tinitus in particular, it has been shown that iterative application of CRT with resting periods among sessions has an ever faster and better therapeutic effect. The specific mechanisms which lead to this are still subject to research. Degenerative diseases such as Parkinson's and Alzheimer induce slow changes in brain networks, leading to deterioration of higher brain function and memory. Lesions and brain disease can transiently reconfigure local excitation-inhibition to an immature state [69]. Tests performed on healthy and stroke patients have shown that the human brain undergoes morphological alterations in response to learning and/or rehabilitation [137, 56, 20]. These alterations can be linked to a trigger of periods of structural plasticity which aim at reinstating a previously healthy or a more mature state of the brain circuitry.

6.2 Implementation of the STN-GPe neural network with STDP and structural plasticity

6.2.1 The Network Model

The network model used in this chapter is described in Manos et al. [103] and consists of two interacting nuclei, the STN and the GPe. The STN neuron population is linked with the cortex in an excitatory manner while the GPe one is linked with the striatum in an inhibitory manner. The connectivity matrix of the STN-GPe network can be expressed as the combination of several sub-networks:

$$W = \begin{pmatrix} w_{ss} & w_{sg} \\ w_{gs} & w_{gg} \end{pmatrix} \tag{6.1}$$

denoting the connectivity matrices for the STN-STN (ss), STN-GPe (sg), GPe-STN (gs), and GPe-GPe (gg) sub-networks, respectively. Each W_{ij} value corresponds to the connection strength between neuron i and j while self-connections are excluded, i.e. $W_{ii} = 0, \forall i$. The neuronal network is a weighted and directed graph, and hence, this matrix is not necessarily symmetric. In Figure 6.1, the basic feature components of the network configuration are shown. The synaptic connections in the STN population are excitatory while in the GPe population they are inhibitory. Red and blue arrows indicate excitatory and inhibitory synaptic connections without plasticity, respectively. The red dashed line represents plastic connections between the STN neurons. The external electrical stimulation exerted to the STN is indicated by the gray solid line as DBS (deep brain stimulation).

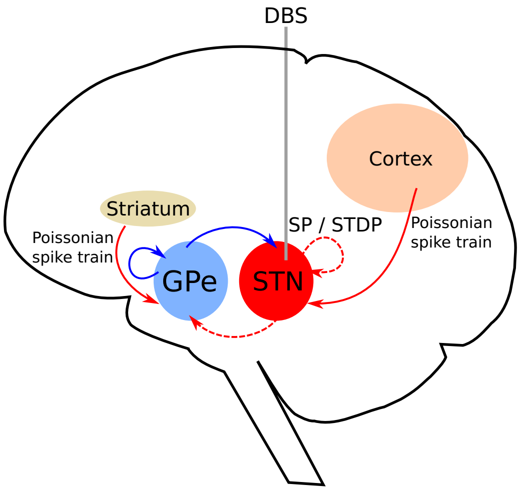


Figure 6.1: Structure of the model network. Excitatory synaptic connections are shown in red while the inhibitory ones in blue. Dashed red lines represent connections within the STN population which are subject to structural and synaptic plasticity. The external electrical stimulation is exerted to the STN population. The STN population receives input from the cortex in the form of poissonian spikes. The GPe population receives input from the Striatum also as poissonian spikes.

6.2.2 Network and connectivity models

The simple network model consists of $1,000$ STN and $1,000$ GPe neurons, while the large model uses 10 times as many neurons on each population. The number of neurons was chosen to be a balance between a biologically realistic and a computationally treatable model. The coordinates of each neuron have been acquired from magnetic resonance imaging (MRI) data taken from a PD patient before DBS surgery, both from the left-brain hemisphere. See [45] for a detailed description of the motivation and construction of the whole model.

The model presented here has 4 different types of synapses, STN-GPe, STN-STN, GPe-STN and GPe-GPe. All parameters are listed in Table (6.1). The STN neuron excitatory connectivity values are based on 11 , Each STN neuron extends connections to 7% of the entire STN population, 700 synapses per neuron and 7×10^6 synapses in total for the large model. Within the STN population synaptic delays are set $\delta_{ss} = 4.0\,\text{ms}$. The connection probability between neurons follows the exponential law $p(x) = e^{-x/c_d}$ with $c_d = 0.5$. Correspondingly, the inhibitory GPe neurons are connected to 1% of other neurons in the population, 100 synapses per neuron and

1×10^6 synapses in total. The delay within the GPe population is $\delta_{gg} = 4.0\,\mathrm{ms}$ and $c_d = 0.63$. The initial synaptic weights for both STN and GPe connections are drawn from a Gaussian probability distribution around a mean value $w_{ss} = 0.0025$ with standard deviation $\delta_{ss} = 0.000125$. The two structures are connected to each other as the STN affects GPe via excitatory input while the GPe extends inhibitory connections on the STN. There is no distance dependence for the connectivity between both populations. Connection probability is 2%, which translates into 200 randomly picked synapses with transmission delay of $\delta_{sg} = \delta_{gs} = 6.0\,\mathrm{ms}$ for connections between the STN and the GPe. Corresponding initial synaptic weights are drawn again from a Gaussian probability distribution around a mean value $w_{sg} = 0.006$ and $w_{gs} = 0.003$ with standard deviation $\sigma_{sg} = 0.0003$ and $\sigma_{sg} = 0.00015$ respectively.
The postsynaptic currents are described with α-function:

$$\alpha(t) = \frac{t - t_k}{\tau_{syn}^2} e^{\frac{t-t_k}{\tau_{syn}}}, t_k \leq t < t_{k+1} \tag{6.2}$$

where t_k denotes the spike time. The total synaptic input current to a postsynaptic neuron i received from presynaptic neurons j is then given by:

$$I_{syn,i}(t) = \sum_j W_{ij}(v_i(t) - v_{syn})\alpha(t), \tag{6.3}$$

where W_{ij} is the synaptic weight between the presynaptic neuron j and the postsynaptic neuron i and v_{syn} is the reversal potential for excitatory or inhibitory connections. Therefore, v_{syn} depends on the type of connected neurons and on whether the connection is excitatory or inhibitory.

6.2.3 The Terman-Rubin Neuron Model

In [103] we used the single-compartment conductance-based Terman and Rubin model [147, 128] to describe STN and GPe neurons. In this model the membrane potential is given by the following equation:

$$c_m \frac{dv}{dt} = -I_L - I_K - I_{Na} - I_T - I_{Ca} - I_{ahp} - I_{syn} + I_{stim} + I_{noise} \tag{6.4}$$

Spiking activity is caused by the sodium (Na^+) and potassium (K^+) ionic currents I_{Na}, I_K. I_T and I_{Ca} describe the low-threshold T-type and high-threshold Ca^{2+} current respectively. I_{ahp} represents a Ca^{2+} -activated, voltage-independent after-hyperpolarization K^+ current and I_L the leak current. In addition, the STN and GPe neurons are influenced by synaptic inputs I_{syn}.

Surrounding brain areas contribute with I_{noise} while there is an external stimulation current I_{stim} (only in the STN population) which models the deep brain stimulation (DBS). Other ionic currents (in pA/μm^2) are described by:

$$I_L = g_L[v - v_L] \tag{6.5}$$

$$I_K = g_K n^4[v - v_K] \tag{6.6}$$

$$I_T = g_T a_\infty^3(v) b_\infty^2(r)[v - v_{Ca}] \tag{6.7}$$

$$I_{Ca} = g_{Ca} s_\infty^2(v) h[v - v_{Ca}] \tag{6.8}$$

$$I_{ahp} = g_{ahp}[v - v_K]\frac{[Ca]}{[Ca] + k_1} \tag{6.9}$$

$$\frac{d[Ca]}{dt} = \Gamma(-I_{Ca} - I_T - k_{Ca}[Ca]) \tag{6.10}$$

STN and GPe neurons are described by similar equations and they differ only in a few parameter values as well as in the form of the low threshold Ca^{2+} current $I_T = g_T a_\infty^3(v)(r)[v - v_{Ca}]$ for the GPe neurons (i.e. the b_∞^2 term is omitted). All the mean parameter values and their units are summarized in Table (6.1). Neurons are not identical, i.e. their reverse potential parameters and ion channel maximum conductances are drawn from Gaussian probability distributions with 10% standard deviation around the mean value. The refractory period is set equal in both STN and GPe neurons at $\tau_{ref} = 3\,$ms.

6.2.4 Noise inputs

Each GPe neuron receives external inhibitory input from the striatum which is described by a constant negative input current of $-7.0\,$pA as well as Poissonian spike trains with a frequency of 40 Hz. STN neurons receive Poissonian spike trains with a frequency of 20 Hz emulating input received from the Cortex. See [45] for more details. Each neuron receives independent random Poissonian spike trains. These noise frequency values are chosen with respect to the respective STN and GPe relative firing rate.

Parameter	STN	GPe	Units	Parameter	STN	GPe	Units
g_L	2.25	0.1	$\mathrm{nS}/\mu\mathrm{m}^2$	g_K	45.0	30.0	$\mathrm{nS}/\mu\mathrm{m}^2$
g_{Na}	37.5	120.0	$\mathrm{nS}/\mu\mathrm{m}^2$	g_{Ca}	0.5	0.15	$\mathrm{nS}/\mu\mathrm{m}^2$
g_{ahp}	9.0	30.0	$\mathrm{nS}/\mu\mathrm{m}^2$	g_T	0.5	0.5	$\mathrm{nS}/\mu\mathrm{m}^2$
v_L	-60.0	-55.0	mV	v_K	-80.0	-80.0	mV
v_{Na}	55.0	55.0	mV	v_{Ca}	140.0	120.0	mV
v_{ss}	0.0	-	mV	v_{sg}	0.0	-	mV
v_{gs}	-	-100.0	mV	v_{gg}	-	-80.0	mV
η_h	500.0	0.27	ms	η_n	100.0	0.27	ms
η_r	17.5	30.0	ms	ξ_h	1.0	0.05	ms
ξ_n	1.0	0.1	ms	ξ_r	40.0	-	ms
υ_h	0.75	0.05	ms	υ_n	0.75	0.05	ms
υ_r	0.2	1.0	ms	τ_{ss}	1.0	-	ms
τ_{gs}	3.3	-	ms	τ_{gg}	-	3.3	ms
τ_{sg}	-	1.0	ms	θ_m	-30.0	-37.0	ms
θ_h	-39.0	-58.0	ms	θ_n	-32.0	-50.0	ms
θ_r	-67.0	-70.0	ms	θ_a	-63.0	-57.0	ms
θ_b	0.4	-	ms	θ_s	-39.0	-35.0	ms
o_m	15.0	10.0	ms	o_h	-3.1	-12.0	ms
o_n	8.0	14.0	ms	o_r	-2.0	-2.0	ms
o_a	7.8	2.0	ms	o_b	-0.1	-	ms
o_s	8.0	2.0	ms	μ_h	-57.0	-40.0	ms
μ_n	-80.0	-40.0	ms	μ_r	68.0	-	ms
k_1	15.0	30.0	ms	γ_h	-3.0	-12.0	ms
γ_n	-26.0	-12.0	ms	γ_Γ	-2.2	-	ms
k_{Ca}	22.5	20.0	ms	Γ	$3.75 \times 10^{-}5$	1×10^{-4}	ms^{-1}
c_m	1	1	$(\mathrm{pF}/\mu\mathrm{m}^2)$				

Table 6.1: Parameters for the definition of STN and GPe neurons.

6.2.5 Spike-Timing-Dependent Plasticity

The weights of the synaptic coupling between neurons in this model are regulated by Spike Timing Dependent Plasticity (STDP). We use the following STDP rule as shown in [45].

$$\Delta w_{ij}(\Delta t) = \begin{cases} \lambda e^{-\frac{|\Delta t|}{\tau_+}}, & \Delta t > 0, \\ -\lambda \gamma e^{-\frac{|\Delta t|}{\tau_-}}, & \Delta_t \leq 0 \end{cases} \tag{6.11}$$

where $\tau_- = 27.5$ and $\tau_+ = 12.0$ are the time constants for the synaptic weight change of depression and potentiation respectively, $\lambda = 0.002$ is the learning rate of the synaptic connection and $\gamma = 1.4$ defines the ratio between depression and potentiation in the synaptic learning rule. Synaptic weights are allowed to take values only on the interval $[0, 0.002]$ in order to avoid non-physiological values.

6.2.6 Structural Plasticity

The primary purpose of the structural plasticity algorithm is to model the neurobiological phenomenon of morphological transformations that a neuron undergoes through time, leading to the creation or deletion of synapses. Using the structural plasticity framework in NEST, a network will self-generate synapses in order to stably achieve a desired profile of electrical activity, a measure that is rather more experimentally accessible than detailed connectivity data. By progressively and slowly changing the connections between neurons in the network and the weight of these connections for all populations simultaneously, the structural plasticity algorithm is able to find a stable configuration with the desired firing rate profile. For details on the structural plasticity model please see Section 2.3.1 and for more details on the implementation please refer to Chapter 4.

Structural plasticity works as an internal controller of the activity in the neural network, smoothly changing the connectivity to reach a target behavior. In this study, Gaussian curves as defined in Equation (2.2) are used to describe the growth rate of connection points for neurons. The basic experimental setup considers $\nu = 0.00008\,\text{synapticelements/ms}$ for the Gaussian growth curve and $\varepsilon = 3.0Hz$ for the synapses between STN and STN neurons and $\nu = 0.00002\,\text{synapticelements/ms}$ and $\varepsilon = 3.0Hz$ for the synapses between STN and GPe neurons. I also setup homeostatic rewiring in the post-synaptic ends of the synapses using a Gaussian growth curve of $\nu = -0.005$ on both STN and GPe synapses with $\varepsilon = 3.0Hz$ and $\varepsilon = 7.0\,\text{Hz}$ respectively.

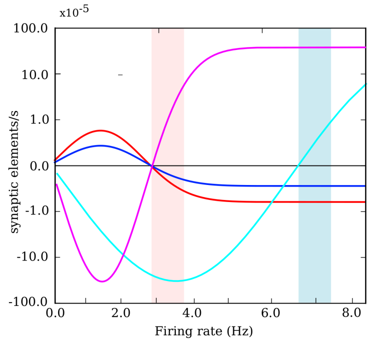


Figure 6.2: Growth rate curve determining the speed of creation and deletion of synaptic elements in the structural plasticity model. The parameters which define the shape of the curve are the growth rate ν which defines the peak of the curve, and the target firing rate ε. The pink band indicates the region where the model operates during the simulations reported in this work for the STN population and the blue band indicates the operation region for the GPe population. The growth curves for the STN to STN synapses (blue), STN to GPe synapses (red), STN rewiring (pink) and GPe rewiring (cyan) are shown in this plot with a maximum growth rate of $\nu = 0.00008\,\text{synapticelements}/\text{ms}$, $\nu = 0.00002$, $\nu = -0.005$, and $\nu = -0.005$ respectively. ε was set to 3.0 Hz for the STN to STN and STN to GPe rules. For the rewiring curves, the value of ε was set to 3.0 Hz for the STN population and 7.0 Hz for the GPe population, matching their target level of activity.

This homeostatic rewiring allows the creation of new post-synaptic contact points which can be associated to freed pre-synaptic contacts during synaptic deletion on each structural plasticity update. The specific curves used in the following experiments can be seen in Figure 6.2. This setup is meant to drive the target STN firing rate towards a non-pathological rate of about 3.0 Hz. Structural plasticity and the Gaussian growth curve are applied only to excitatory connections from and to the STN population, and to excitatory connections from STN to the GPe population as seen in Figure 6.2. The growth rate has only positive values in this case. The curve specifies that when neurons have a firing rate higher than 3.0 Hz, synapses will be deleted; if neurons have a firing rate lower than 3.0 Hz, synapses will be created. In this case, I am working in a regime of about 3.6 Hz at the moment structural plasticity is enabled after each stimulation cycle, which is higher than the target firing rate. This means that during the simulation, synapses will be slowly deleted until the desired 3.0 Hz target is reached.

For these simulations I have artificially accelerated the changes in the structure of the neural network in order to make feasible the simulations within reasonable times. Structural plasticity was integrated to the topology module in NEST, in order to be able to work with layers and the creation functions particular to this module. The model also makes use of STDP as synaptic plasticity algorithm. The selective activation and deactivation of both the structural and synaptic plasticity algorithms was required in order to allow a smooth interaction between them.

6.2.7 External Coordinated Reset stimulation

The external invasive DBS electrical CR stimulation performed to the STN population was modeled as a signal of short biphasic current or voltage pulses. This is the same as previously used in [45] (see ”Stimulation input” for original motivation). In more detail, a current pulse $P(t)$ for stimulation of κ,

$$P(t) = \begin{cases} \kappa, & t_l \leq t < t_l + \omega, \\ -\kappa/p_s, & t_l + \omega \leq t < t_l + \omega(1+p_s), \\ 0, & else \end{cases} \tag{6.12}$$

t_l denote the onset times of the current pulses, κ is the amplitude, and ω is the width of the cathodal pulse. p_s determines the duration and amplitude of the charge-balancing anodal pulse part which prevents any permanent charge transfer into the neuronal tissue that could possibly damage the tissue. In this work, we use a fixed electrode position at the center of the STN population which was found to be most adequate for optimal CR stimulation performance in Ebert et al. [45]. In order to model the electric field produced by the external stimulus, we used the setup of Medtronic DBS lead model 3389 (broadly used in clinical trials) which has four separate cylindrical contacts made of a Pt-Ir alloy with a typical length of 1.5mm [28]. We used an explicit equation which approximates adequately the overall distance dependent decay of stimulation strength as used in Ebert et al. [45]:

$$S(d_{il}) = \frac{1}{d_{il} l_c \sqrt{1 + 4(\frac{d_{il}}{l_c})^2}} \tag{6.13}$$

where d_{il} is the distance between neuron i and the location of the stimulation contact l and l_c is the length of the electrode contacts. We prohibit possible neuron coordinates within a cylindrical volume with 1.4 mm diameter around the electrode axis. We use 4 stimulation sites and all neurons receive input from all sites. During each stimulation period of duration 125 ms, each site is activated via an electric burst only once and not at the exact same onset time with any other site. This order of activation varies randomly in every cycle. Three cycles of CR stimulation

(ON-cycles) are followed by two cycles without stimulation (OFF-cycles). The stimulation period within a burst is set at $T_p = 7.69\,\text{ms}$, the CR stimulation amplitude $\kappa = -3.3$ is, the width of the cathodal pulse at $\omega = 200\,\mu\text{s}$ and the duration and amplitude of the charge-balancing anodal pulse at $p_s = 8$. For a detailed description of the CR stimulation setup see [45].
The synaptic weights, being affected by the STDP and the different intrinsic periods of the neurons, change dynamically in time.
In order to measure the degree of the neuronal synchronization within the neuronal population at time t we use the order parameter R:

$$R(t) = |N^{-1} \sum_j e^{i\varphi_j(t)}|, \tag{6.14}$$

where i here denotes the unit imaginary unit $\sqrt{-1}$, $\varphi_j(t) = 2\pi(t - t_{j,m})/(t_{j,m+1} - t_{j,m})$ for $t_{j,m} \leq t < t_{j,m+1}$ is a linear approximation of the phase of neuron j between its m^{th} and $(m+1)^{th}$ spikes at spiking times $t_{j,m}$ and $t_{j,m+1}$. The order parameter R measures the extent of phase synchronization in the neuronal ensemble and takes values between 0 (absence of in-phase synchronization) and 1 (perfect in-phase synchronization).

6.2.8 Using the model to investigate the role of structural plasticity in CRT

All simulation protocol used in this work start with the model in a desynchronized state with down regulated synapses. Simulation were performed using a modified version of NEST which is able to work with structural plasticity and the topology module. This version also allows the user to disable STDP in selected synapses and enable structural plasticity in the same simulation. Since the two types of plasticity take place at very different time scales, I turn off the STPD during the periods when structural plasticity is enabled, assuming its impact is negligible and no external stimulus is delivered during that time. In the NEST simulations I present in this work, the update interval of the structural plasticity algorithm was set to 1000 simulation steps. During this interval, synaptic elements grow and recede according to the firing rate of the neurons at the same time scale as the electrical activity is calculated, but it is only when the time interval is complete that the actual connections between neurons are updated. The firing rate of each neuron is calculated by NEST, using a low pass filtering technique. For more details on how this calculation is performed as well as the impact of the update interval and other structural plasticity

parameters in the evolution of the simulations (see Nowke et al. [114]).
All runs were performed on the JUWELS supercomputer in the Jülich Supercomputing Centre.

Protocol 1: Simulations only using STDP with 2 stimulation cycles

1. *Step* 1*:* Apply sufficiently strong periodic stimulation (PS) to induce a kindling. By doing this, the network is shifted to a strongly (synaptically) connected, synchronized and stable state. This periodic stimulation is performed for 2.5 min of biological time.
2. *Step* 2*:* Once this stimulation is over, deliver CR stimulation causing a desynchronization and reducing the order parameter sufficiently well. This stimulation takes 10 min of biological time.
3. *Step* 3*:* Start a 10 min period of CR-off where the network relaxes to a desynchronized state with down-regulated synaptic weights. These three steps are performed with STDP activated.
4. *Step* 4*:* Start a longer waiting phase which is intended to represent days of biological time between stimulation sessions. In the simulation, this takes 20 min.
5. *Step* 5*:* Repeat Steps $1-3$.

Results of this initial simulation can be seen in Figure 6.3 As it can be observed, the model behaves exactly the same during the second phase of synchronization and desynchronization. There is no trace of long term changes induced to the network during the first CR stimulation.

Protocol 2: Simulations using structural plasticity with 2 stimulation cycles

1. *Step 1:* Perform Steps $1-3$ in the same way as we did in Protocol 1.
2. *Step 2:* When the longer waiting phase is started, STDP is turned off and structural plasticity is turned on.
3. *Step 3:* Repeat Steps $1-3$ from Protocol 1 again.

Results of the second protocol can be seen in Figure 6.4 In this second protocol it becomes visible that, when structural plasticity is enabled, the effects of the second CRT stimulation are faster and better than the first time. Structural plasticity reinforces the desynchronization of the network achieved by CRT and STDP by means of pruning synapses.

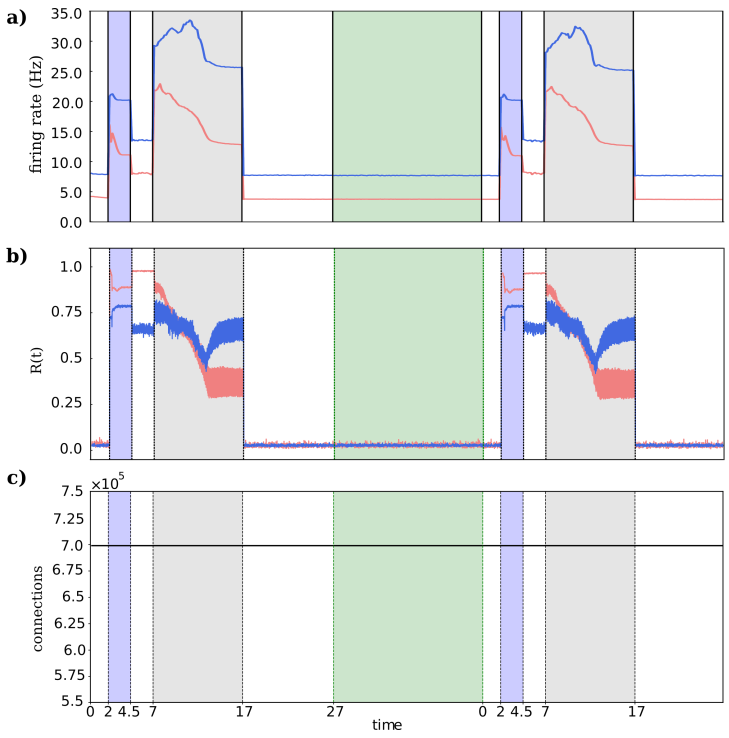


Figure 6.3: Synchronization, firing rate and connections for 2 long CR sessions without SP. a) Spiking rates for STN and GPe neurons respectively. Light blue bands denote the Periodic Stimulation (2.5 min) intervals and light gray the CR-ON stimulation periods (10 min). STDP is active throughout the whole-time evolution. b) Time evolution of the order parameter $< R >$ averaged over a sliding window (100 ms) for STN (red solid line) and GPe (blue solid line) neurons. c) Number of total connections in the STN population during the simulation.

Protocol 3: Simulations using structural plasticity with 3 stimulation cycles

1. *Step 1:* Perform Steps $1-2$ in the same way as in Protocol 2.
2. *Step 2:* Perform Step 3 in the same way as in Protocol 2 but with a reduced CR time of 6 min.
3. *Step 3:* Repeat Steps $1-2$ again from Protocol 2.
4. *Step 4:* Perform Step 3 in the same way as in Protocol 2 but with a reduced CR time of 5 min.

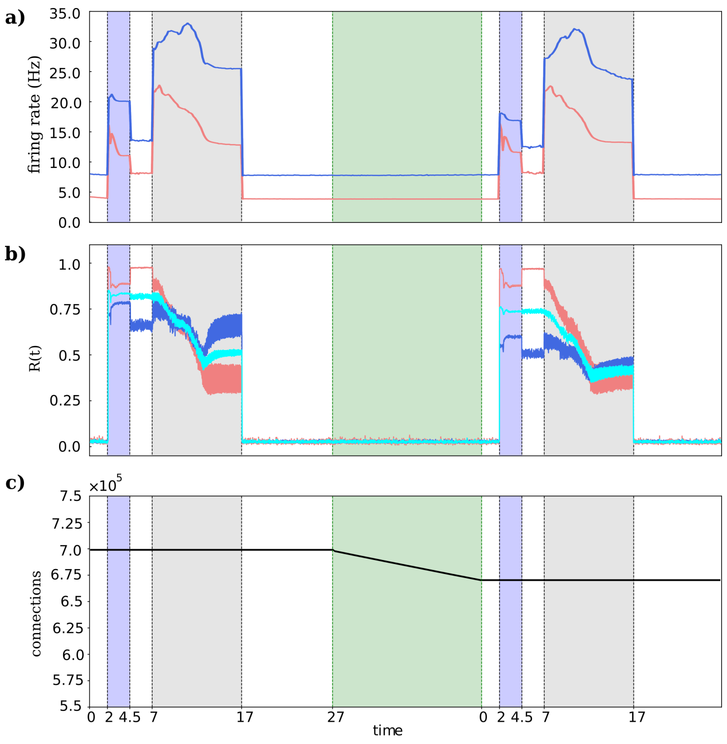


Figure 6.4: Synchronization, firing rate and connections for 2 long CR sessions with SP. Light blue bands denote the Periodic Stimulation (2.5 min) intervals, light green bands denote periods where only SP was active, and light gray the CR-ON stimulation periods (10 min). a) Spiking rates for STN and GPe neurons respectively. b) Time evolution of the order parameter $< R >$ averaged over a sliding window (100 ms) for STN (red solid line) and GPe (blue solid line) neurons, as well as the whole network averaged value (cyan solid line). c) Number of total connections in the STN population during the simulation.

Results of the third protocol can be seen in Figure 6.5 In this third protocol the effects of structural plasticity are even more visible in the third stage of synchronization and therapy. It is also visible that CR only requires 6 min the second time and 5 min the third time to desynchronize the network when SP is used (panels a-c), while this is not possible in the case without SP (panels d-e).

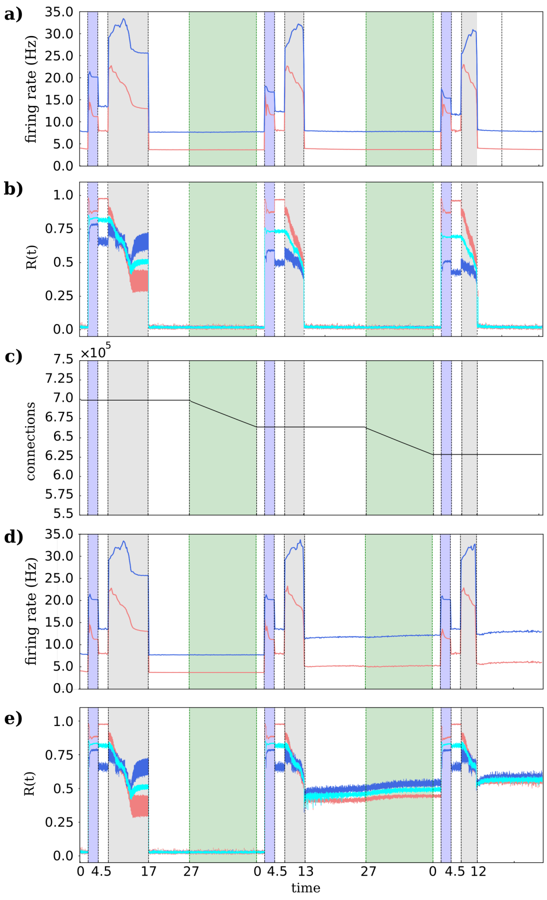


Figure 6.5: Synchronization, firing rate and connections for 3 CR sessions with SP compared to the non SP scenario. Light purple bands denote the Periodic Stimulation (2.5 min) intervals, light green bands denote periods where only SP was active, and light grey the CR-ON stimulation periods (10 min, 6 min, and 5 min correspondingly) a) Spiking rates for STN and GPe neurons respectively **with SP**. b) Time evolution of the order parameter $< R >$ averaged over a sliding window (100 ms) for STN (red solid line) and GPe (blue solid line) neurons, as well as the whole network averaged value (cyan solid line) **with SP**. c) Number of total connections in the STN population during the simulation. d) Spiking rates for STN and GPe neurons respectively **without SP**. e) Time evolution of the order parameter $< R >$ averaged over a sliding window (100 ms) for STN (red solid line) and GPe (blue solid line) neurons, as well as the whole network averaged value (cyan solid line) **without SP**.

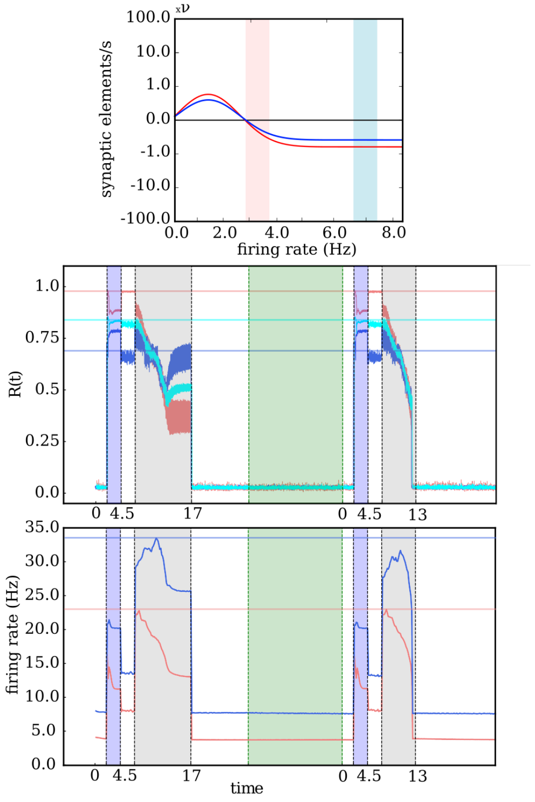


Figure 6.6: Growth curve, synchronization and firing rate for a simulation where all excitatory synapses from STN to STN and STN to GPe are regulated by structural plasticity. For growth curves, the same color coding is used as in Figure 6.2. In the growth curve plots, blue stripes define the operation region of the GPe population while the pink stripes define the operation region of the STN population. In the synchronization and firing rate plots use the same color coding as in Figure 6.4. In these plots light blue, red, and cyan horizontal lines are used to compare values between the first and the second CR stimulation sessions.

6.2.9 Parameter exploration and tunning

The long lasting effects of structural plasticity in this model regarding the enhanced outcome that CRT can achieve are visible in protocols 2 and 3. However, as already explained in previous chapters, the algorithm has several parameters which affect the speed of creation and deletion

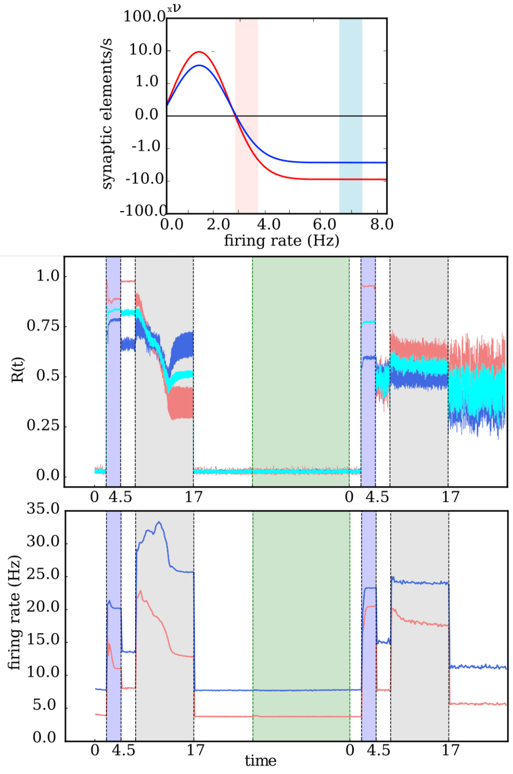


Figure 6.7: Growth curve, synchronization and firing rate for a simulation with high growth and deletion rate (1×10^{-3}) for the STN to STN synapses. Same color coding is used as in Figure 6.6

of synapses. Figures 6.6 to 6.9 show examples of the behaviour of this model with different parameters of structural plasticity. These examples were extracted among 250 parameter combinations explored with this model. Each Figure has a different growth curve and shows particular characteristics. Figure 6.6 shows simulations where the synapses between the STN and STN neurons as well as the ones between the STN and GPe neurons are regulated with structural plasticity. As it can be seen, similar as the example in protocol 3, the system attempts to reach the target activity regime. However, the effects are less visible as protocol 2 and there is low rewiring, and mostly creation and deletion of synapses. It is also visible that in the second phase, the CR

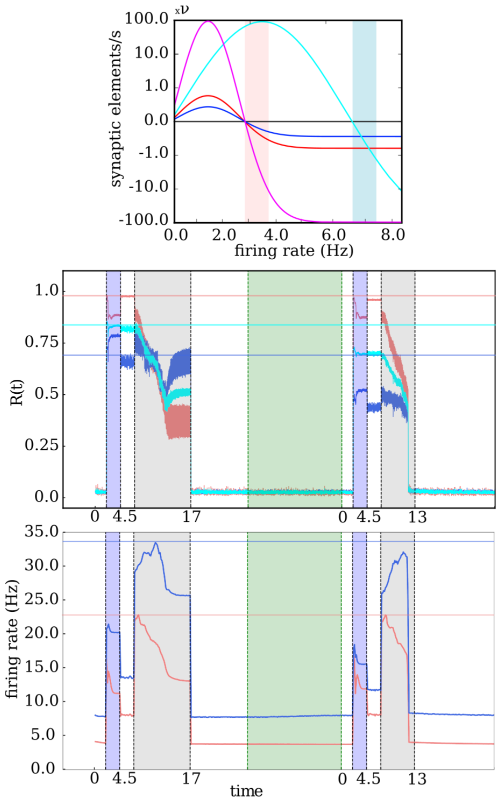


Figure 6.8: Growth curve, synchronization and firing rate for a simulation with same configuration as protocol 2 but higher rewiring values. Same color coding is used as in Figure 6.6

stimulation does not have the same desynchronization effect as in the first stimulation phase, making it possible to reduce the time required for desynchronization after structural plasticity takes place. The results in Figure 6.7 reflect the behaviour of the network for a growth curve with a rate of 1×10^{-3} and a target firing rate of 3.0 Hz. Here the deletion and creation of synapses takes place very fast, moving the system out of an stable regime and making it impossible to desynchronize after the first structural plasticity period. The system also shows artifacts in the interface between the STDP and structural plasticity active periods. Figure 6.8 shows an example

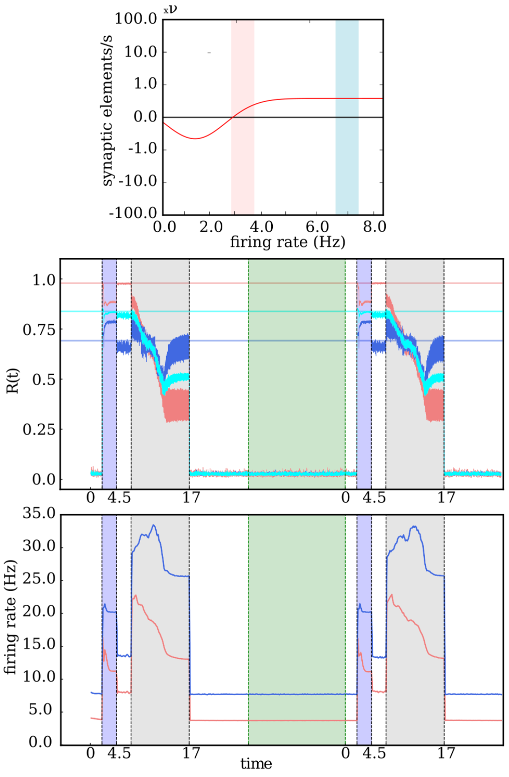


Figure 6.9: Growth curve, synchronization and firing rate for a simulation with low growth and deletion rate. Same color coding is used as in Figure 6.6

similar to protocol 2 but with a higher rewiring factor. The system shows an stronger susceptibility to desynchronziation in the second CR stimulation phase, but a plateau in the GPe population forms in the middle of the CR stimulation. In other simulations with even stronger rewiring, it was possible to see that this plateau impacts the network's internal balance and reduces the effectiveness of the CR stimulation. Finally, in Figure 6.9 the system is simulated using a growth rate of -8.0×10^{-5}. Here we can see an inverse effect, where new connections are created and reinforce the susceptibility of the network to getting into a synchronized state and remaining there. This means that the CR stimulation period takes longer time to desynchronize the network.

6.3 Discussion

This is the first computational study investigating the impact of SP on the outcome of CR stimulation. We implemented SP in a STN-GPe network with plastic STN-STN synapses governed by STDP. In the absence of structural plasticity, repeated administration of PS-CR sequences leads to reversible transitions between desynchronized and synchronized states Figure 6.3. By the same token, delivering PS-CR sequences with CR epochs of insufficient duration causes transitions between synchronized states of different amount of synchrony. To date, changes of susceptibility to CR stimulation after stimulation-free pauses have not been observed in computational studies. However, such effects were previously observed in a clinical study with acoustic CR stimulation [146], where therapeutic responses of tinnitus patients to CR stimulation were particularly strong after a pre-planned 4-week treatment pause. In general, effects of this kind may allow the further reduction of the total stimulation duration; in an example such as DBS, this could further reduce the rate of side effects. In our model, comparable memory-type of treatment effects come into play when structural plasticity is activated (Figure 6.4-Figure 6.5). On one hand, the PS epochs in the PS-CR sequences may serve as a standardized model process accounting for detrimental influences on patients during stimulation-free intervals, leading to a re-increase of symptoms (see [146]). On the other hand the PS epochs may be considered as standardized probes, enabling the testing of the network's resistance to synchronizing stimulation protocols.

Assuming that the target firing rate is within a range characterizing a healthy STN firing state, structural plasticity continuously rewires, modifies the connectivity and down regulates the firing rate by deleting excitatory synapses. Consequently, during the PS epoch the STN re-synchronization is less pronounced and does not reach the initial levels observed by the first PS epoch (Figure 6.4). The amount of synchronization of the GPe network, which provides an inhibitory output, during and directly after the PS epoch does not change significantly compared to the first PS epoch. However, during the subsequent CR epoch the desynchronization of STN and GPe are both accelerated in a similar manner compared to the first CR epoch. Accordingly, a long-lasting desynchronization of STN and GPe can be achieved by means of a considerably shorter CR epoch (e.g., half the duration).

An important prerequisite of the structural plasticity effect is that after the CR epoch the STN firing rate relaxes to a value sufficiently close to the structural plasticity target firing rate. This enables structural plasticity to take over efficiently and down-regulate the number of excitatory connections within the STN, together with the firing rate using the homeostatic rule. The choice

of a Gaussian curve in the structural plasticity model employed in this study was motivated by experimental data [5, 96] which suggests that the generation of new spines starts slowly when the activity in the neuron is low and progressively increases to a maximum. Then it starts to decay until a homeostatic equilibrium is reached [22]. During development the degree of plasticity during critical periods also increases slowly until a maximum level, and then decreases until a degree of maturity is achieved (see box 1 in [69]). It is interesting to note that a reduction by 5.76% of STN-STN and STN-GPe connections during the structural plasticity epoch is sufficient to increase the network's resistance to PS and also its increased susceptibility to CR stimulation; the SP effects accumulate, even though intersected by a PS-CR sequence.
The structural plasticity-mediated effect of increased resistance to PS and subsequent increased susceptibility to CR stimulation depends on the target firing rate. If a higher frequency, say 3.6 Hz instead of 3.0 Hz, is used as target firing rate for the homeostatic structural plasticity mechanism then the deletion of excitatory synapses is less pronounced; in turn, the network's SP-induced resistance to PS and its subsequent increased susceptibility to CR too (Figure 6.7)). For the structural plasticity algorithm, a Gaussian growth curve with target firing rate of 3.0 Hz was chosen. This target firing rate was chosen because it is slightly below the final firing rate of the neurons after the CR therapy. The theory behind this selection is that CR teaches the network a new activity regime, which structural plasticity reinforces. Simulations were performed with a variable growth rate from 0.00001 to 0.0001 synapticelements/ms. The parameter search was performed in order to match the experimental results. The constraints to the selection of suitable network transitions included no instabilities of the network activity between CR sequences. It was also visible that structural plasticity in both STN-STN and STN-GPe synapses was required in order to achieve stable networks, and a good balance between the creation and deletion of both types of synapses is also highly relevant to reproduce experimental results. The main motivation and rationale were to design a SP mechanism whose time evolution would cause sufficient synaptic deletion in order produce an impact on the macroscopic measurement (order parameter) within a computationally feasible time
In the brain there are several plasticity rules which take place simultaneously and the interaction between them, their specific detailed mechanisms and different roles are still not fully understood. However, increasing experimental evidence shows that synaptic plasticity takes place on time scales that may range from seconds to minutes and up to hours (see e.g. Zenke and Gerstner [169]). In contrast, structural plasticity is a much slower process which has been identified to take place across longer time frames, i.e. from several hours through to days, weeks and months

(see e.g. Van Ooyen and Butz-Ostendorf [155]). Harnack et al. [61] explain the homeostatic control as an internal mechanism of neurons which allows them to keep a healthy operating regime. Their analytical framework is based on control theory and explores possible and adequate conditions for stability in recurrent networks which can be directly linked to different mechanisms of plasticity such as the ones were explored in this chapter. In this study and the novel NEST based code, the power of structural and synaptic plasticity are combined to provide a modeling test environment that captures memory-like phenomena of susceptibility to CR stimulation. From the modeling and simulation perspectives, there are several challenges related to this setup; the most significant is that simulations of detailed networks are computationally demanding, especially when the two types of plasticity are involved. This is mainly due to the variations in the transmission and generation of spiking events in the network, as well as the additional calculations that take place at the synapse level. Simulations of the size used in this chapter can have a simulation to real-time factor of about $10-20$. This means that simulations of 1 s take $10 to 20$ s to be simulated computationally. Accordingly, exploring slow SP-induced changes of a simulated network within 1 month of biological time, could take a normal simulation of up to several (20) months duration. Obviously, such a scenario is significantly resource-intensive. In contrast, using NEST simulations, the effects of structural plasticity can be accelerated so that simulations can be reduced from months to a couple of hours. With this solution, I am able to simulate at feasible computing times both the synaptic plasticity, which takes place during PS and CR epochs, followed by a stabilization period and the subsequent activation of SP, running at reasonably high speed by means of an accelerated model. During this accelerated simulation period, I disable synaptic plasticity in order to avoid inconsistencies that would be induced due to the fact that the model operates at different time scales.

In a previous computational study in a neural network with STDP, but no structural plasticity, it was shown that spaced CR stimulation, i.e. CR stimulation intermingled with sufficiently long stimulation-free pauses, may significantly improve efficacy; this is particularly important if CR stimulation is delivered at otherwise ineffectively weak intensities [121]. That study, together with our current study, illustrates the importance and active role of pauses in therapeutic processes employing plasticity principles. Stimulus-free intervals should not just be considered periods without intervention. Rather they may be an integral part of a therapeutic process enabling and potentiating stimulus effects which should be adequately addressed by dose-finding studies. By combining both types of plasticity it is possible to explore the immediate stimulation-induced effects of synaptic plasticity as well as the long-term effects mediated by structural plasticity

within a single consistent, efficient, and tractable simulation model. The model described in this work provides a simulation test space for neuroscientists to evaluate and study the effects of plasticity and stimulation as well as their interactions for therapeutic purposes.
Future studies could, for example, be devoted to the development of desynchronizing stimulation protocols that are adapted to specific ranges of target firing rates. To this end, it is crucial to understand stimulation-induced changes of the firing rate. In fact with very few exceptions (see e.g. Lysyansky et al. [101]), the vast majority of computational studies devoted to CR stimulation to date have not focused on stimulation-induced changes of the firing rate. This study highlights that not only the CR epochs, but also pauses in between CR epochs, may be crucial for the long-term stimulation outcome. CR stimulation and, in general, all kinds of efficient desynchronizing stimulation protocols may initiate STN-GPe circuits in favorable states, so that SP makes networks more susceptible to desynchronizing stimulation over time.

7 Structural plasticity as a connectivity optimization algorithm

In the previous chapter I discussed an application of the structural plasticity model implemented in NEST in the field of clinical neuroscience. In Chapter 3 I presented a set of algorithms frequently used in machine learning and discussed similarities to how structural plasticity works. Chapter 5 also supports the potential of using structural plasticity as an optimization algorithm for taking a neural network from an initial activity regime to another, following homeostatic rules. In this chapter, I use structural plasticity to find network structures which lead to functional regimes of interest for neuroscientists in large scale simulations. I leverage the potential of structural plasticity to find network architectures by optimizing their connectivity against a specific measure i.e. firing rate. The parameter space which can be used to define dynamic point-to-point connectivity in a neural network is vast, and increases quadratically ($O(n^2)$) with respect to the number of neurons without further constraints. In this chapter I exploit structural plasticity to find static structures and also combine it with other machine learning algorithms (as the ones described in Chapter 3) to find general plasticity rules which can take a neural network from an initial immature functional state to a final mature functional state. This optimization of plasticity rules can also help understand critical relationships in the evolution of brain structures during learning, healing or development. This approach allows us investigate general connectivity rules that can give more insight on how the brain achieves and preserves its function.

7.1 Definition of the structural plasticity model

In the following I will describe the specifics of the structural plasticity model I use for the simulations in this Chapter. A Gaussian curve, as shown in Figure 2.2, is an example of a homeostatic rule and describes the growth rate of connection points for neurons as a function of the neurons firing rate as described in Section 2.3.1. In this Chapter, I make use of homeostatic curves of this type to guide the structural plasticity model and will make reference directly to the mean firing rate (as in Chapter 5) to guide the optimization algorithms. As already explained in Chapter 4, the parameters defining the growth and decay of synapses are the minimum firing rate required to start generating synaptic elements η, the growth rate ν, and the target firing rate ε. Modifying these values alters the way connections are created and deleted in the network.

In Chapter 5 I presented an interactive steering and visualization tool for structural plasticity, with the objective of exploring the complex parameter space of potential homeostatic rules through time. The tool combines interactive steering and visualization interfaces, middle-ware for dynamic software component connection, and a live NEST simulation implementing structural plasticity dependent upon a large-dimensional parameter set. This development was driven by the need to rapidly reach stable configurations of connectivity with the supervision of a human controller. The tool allows for the visualization of the trajectories the system undergoes during simulation by showing the changes in the observable states of the network, namely, the activity and connection properties of the network. It also allows the user to change the shape of the homeostatic curve and visualize the impact of the changes in connectivity while the simulations are running. The topic of this chapter is how to use an automated version of this process and allow an optimizer to guide the search. The optimizer is integrated into the interactive steering and visualization tool in such a way that human monitoring is still possible during automated explorations.

In this chapter, I combine the neurobiologically inspired structural plasticity algorithm (which I will refer also as the *optimizee*) with a machine learning supervising algorithm (which I will refer to as the *optimizer*). The aim is to allow the optimizer to guide the broad search of relevant parameters to answer our scientific questions while, at the same time, imposing neurobiological restrictions using the structural plasticity rules below it. This will be done by applying the optimizer to the homeostatic rules which guide the optimizee. By generating a series of homeostatic rules which enable certain transitions of interest within the system, I aim at obtaining a better understanding of how homeostatic changes in the network affect its activity.

7.2 Adaptive optimization of structural plasticity in spiking neural networks

As stated before, the goal of this chapter is to find homeostatic rules for the structural plasticity algorithm which provide a range of expected trajectories for the dynamics of the neural network, using adaptive algorithms. In order to achieve this goal, I used a meta-learning framework in combination with simulations of spiking neural networks in NEST. Learning to learn [151] is a specific solution for acquiring constraints to improve learning performance. In machine learning, a learning program is trained against samples from a single task with a well-defined performance measure in order to improve its performance against new samples from that same task. As discussed in the previous chapter, meta-learning improves the program's performance at training against samples from a new task from a family of tasks. This is achieved by optimizing the hyperparameters (such as a set of synaptic plasticity values) of the learning system across the family of tasks. This approach begins by training against a learning task with a well-defined performance measure and measuring the learning rate or absolute performance after a fix period of training. Depending on the fitness of the resulting training run, the training is then repeated for new tasks from the family using a variation of the resulting hyperparameters from the previous generation and learning is measured again. Over many generations, meta-learning can occur: the network has been trained to optimize the learning rate for a family of tasks. The learning to learn concept can be decomposed into an inner loop, where an optimizee program (such as a deep learning network or a simulation of a biological neuronal network) learns specific tasks and an outer loop where an optimizer program searches for generalized optimizee parameters (hyperparameters) that learns to improve the optimizee's performance over distinct tasks as measured by a fitness function (see Figure 7.1). The performance of this outer loop depends on the performance of the optimizer and the number of simultaneously cooperating optimizees.

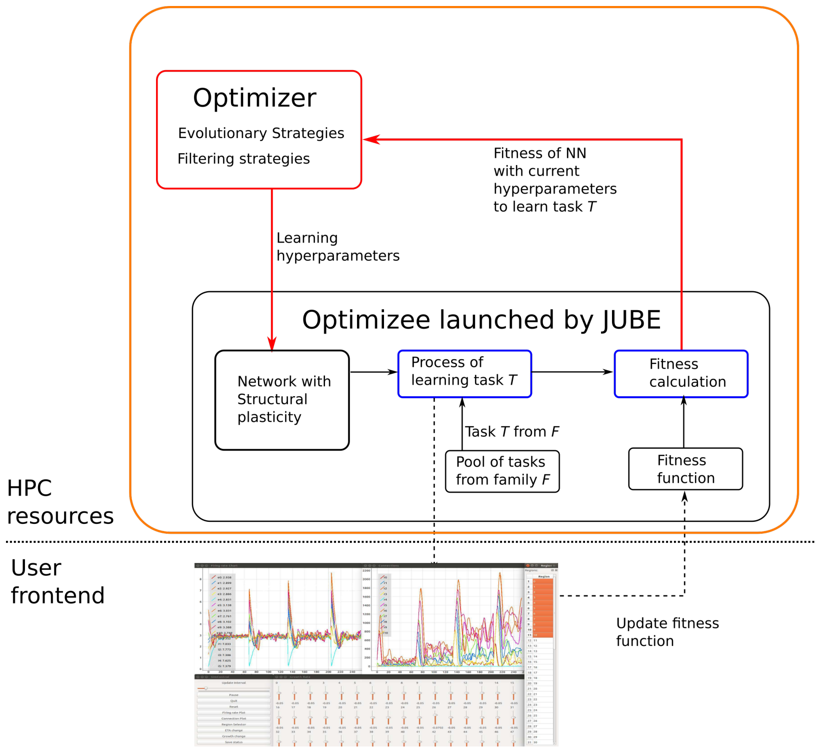

Optimizer
Evolutionary Strategies
Filtering strategies
Fitness of NN with current hyperparameters to learn task *T*
Learning hyperparameters
Optimizee launched by JUBE
Network with Structural plasticity
Process of learning task *T*
Fitness calculation
Task *T* from *F*
Pool of tasks from family *F*
Fitness function
HPC resources
User frontend
Update fitness function

Figure 7.1: Learning-to-learn loop: The optimizee is an ensemble of machine learning instances over sets of hyperparameters and training samples from tasks; Optimizer observes the ensemble and evolves the parameter sets to optimize learning generalization.

The software framework that I used for this project, L2L [142], is an implementation of the learning to learn concept. It is written in Python and in this project I have modified it and merged with JUBE [100] in order to work on the supercomputers as a hyperparameter optimization framework for HPC. The software was tested and refined to be deployed on the JURECA supercomputer in the Jülich Supercomputic Centre in interactive and batch modes. The software is simple to use and allows several instances of the optimizee to be executed with different parameters in a embarrassingly parallel fashion. Each job run provides a parallel set of initial states, from which the search algorithm will iteratively modify the parameters following a fitness rule. This fitness rule is different for each network and tightly linked to the expected transitions

in its dynamics.
The optimization algorithms used in the outer loop were *evolutionary strategies* , *gradient descent* and *simulated annealing*. Please refer to Chapter 3 for more details about the aforementioned optimization algorithms. By keeping track of changes provided to the structural plasticity algorithm and by generating snapshots of the connectivity at given intervals, partial results obtained from previous completed or suspended runs can be used as a starting point for a new job.

7.2.1 Simple network with two populations

The first network explored using hyperparameter optimization is a simple two populations network with 1000 neurons, 80% excitatory and 20% inhibitory. The network is completely disconnected at the beginning of the simulation and structural plasticity is allowed to create both excitatory and inhibitory connections following a base homeostatic rule. Delays and weights are fixed for the connections created.
The optimization task family consists on finding connectivity configurations which lead to defined excitatory and inhibitory firing rates between 4 Hz - 6 Hz and 18 Hz - 22 Hz correspondingly. A test task was defined in order to measure the global learning of the system. This task involves taking the excitatory population to have a mean firing rate of 5 Hz and the inhibitory population to a mean firing rate of 10 Hz. The network is evolved for 200 cycles of 2 s of biological time. Given that different parameters of the growth rate can result in different simulation times, the maximum simulation time allowed was set to 20 min. Simulations which did not finish on time were penalized by the fitness rule. At the end of each cycle, the firing rate of all neurons is sampled. The structural plasticity update step is setup to 1 s. Gradient descent, simulated annealing and cross entropy were chosen as optimizer algorithms in the outer loop. Internally, structural plasticity solves the multi-objective problem of finding a suitable structural configuration of the network which takes each population to its target activity rate. In order to assess the performance of each instance (named as individual in the following) of the network with specific parameters, a multiobjective fitness rule was derived. Several compound fitness measures were tested for the multi-objective optimization. A weighted measure which gives more importance to the fitness of the excitatory than the inhibitory The fitness of each individual is formed 80% of the inverse of the error between the average firing rate of the excitatory population and its target firing rate, and 20% of the inverse of the error between the average firing rate of the inhibitory population and its target firing rate.

Figure 7.2a, Figure 7.2b, and Figure 7.2c show results of the hyperparameter optimization process using the different outer loop algorithms. Here it is important to notice that each dot in the plots represents a full simulation with a particular set of parameters on a random scenario defined by the family of tasks. All outer loop optimization algorithms find that the lower right corner of the selected parameter space is of higher fitness. The parameter combinations get particularly higher fitness when the ration between excitatory and inhibitory growth is between 4 and 5 to 1.
It is possible to see that the simulated annealing algorithm performs a more homogeneously distributed search on the space. This has the advantage that local minima are easier to circumvent. The algorithm still focuses its search efforts in the lower right corner of the parameter space but uses some of the computational resources on areas of low interest. In the case of structural plasticity, simulated annealing is a good candidate to perform initial explorations of vast parameter spaces which provide a good idea of the shape of the space and where areas of interest are located. As an opposite, the cross entropy algorithm is very effective in finding the area of interest with a low amount of resources. This makes best use of computational resources but might miss other interesting areas of the parameter space to be explored. Finally, the gradient descent algorithm shows a behaviour in the middle between simulated annealing and cross entropy in terms of exploration and focus. In all three algorithms we can see that there are parameter combinations outside of the lower right corner which have high fitness. The random nature of the structural plasticity algorithm and the task definition can also cause variability in the performance of the model and thus make the exploration more challenging.
In Figures 7.2a to c the space to be explored was defined within biologically realistic boundaries. Figure 7.2d shows the hyperparameter optimization being performed in a free parameter space in order to show the natural tendency of gradient descent without pre-imposed knowledge. Here it can be seen that the gradient descent algorithm tends to explore an area of the parameter space where inhibitory growth is negative and excitatory growth is positive. This result support other empirical findings reported in Chapter 4, Chapter 5, and Chapter 6 where it can be observed that negative growth rates for inhibitory synapses support the generation of stable structures in the networks.
Figure 7.3 shows the best individual for generation 0, 5, 10 and 15 using gradient descent. As it can be observed, the ability of the network to reach the desired activity gets better as the gradient descent algorithm finds more effective structural plasticity growth parameters.
All experiments were performed on the JURECA supercomputer of the Jülich Supercomputing Centre in Jülich, Germany. Job allocations for this subproject consisted on 51 nodes, deploying 1

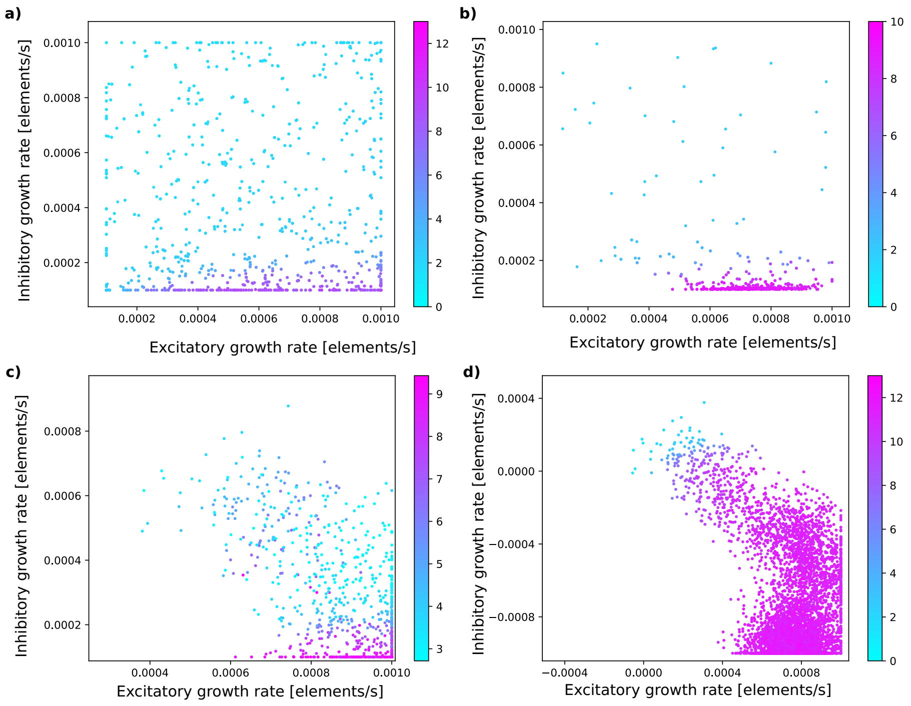


Figure 7.2: Results of the parameter space exploration for the simple two populations network using a) simulated annealing, b) cross entropy, and gradient descent within a c) restricted and an d) unrestricted parameter space. Each dot in the plot represents a position in the parameter space described by the inhibitory growth rate on the y-axis and the excitatory growth rate in the x-axis. The color of the dot represents the fitness of the structural plasticity algorithm using an specific parameter combination to reach the desired activity regime. The fitness values are color coded from cyan to pink using the color bar on the right.

outer loop optimization instance and 50 parallel instances of this network per generation making full use of the 24 cores per node in JURECA. Each generation was forced to end after 20 minutes of simulation. This enforcement was necessary because some parameters make the simulations take longer. Simulations which did not finish on time were penalized by the fitness rule.

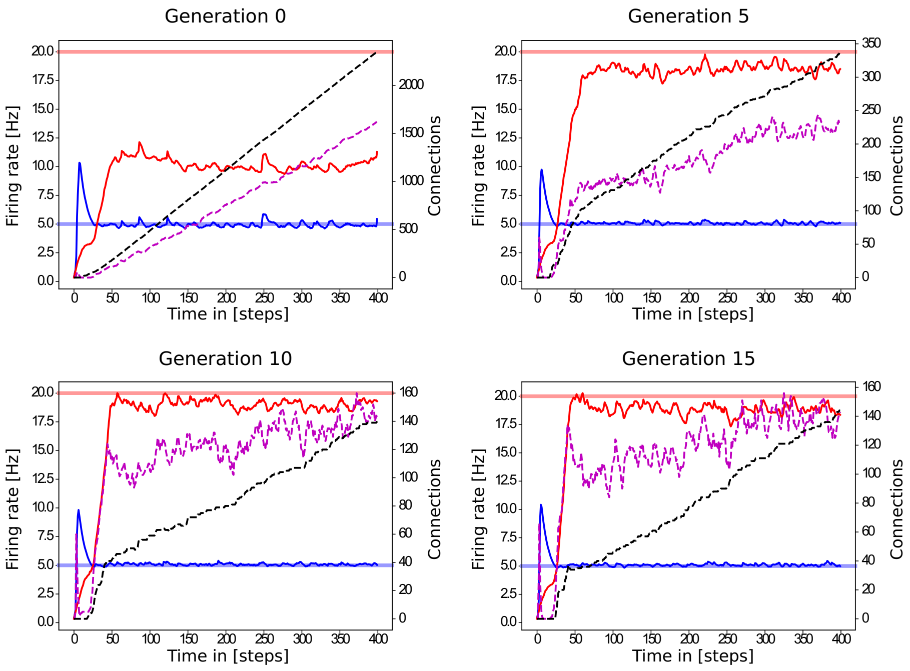


Figure 7.3: Performance on the test task of the best individual using gradient descent for 5, 10, 15 and 20 generations. Blue and red solid lines represent the average firing rate of the neurons in the excitatory and inhibitory populations accordingly. The purple and black dotted lines represent the total connections created by the excitatory and inhibitory populations accordingly. The light blue and red horizontal bars represent the desired target firing rate for the excitatory and inhibitory populations accordingly.

7.2.2 Cortical microcircuit

In this second use case, we create a four layer network based on the model of the cortical microcircuit proposed by [122]. Each layer contains one inhibitory and one excitatory population of leaky integrate and fire neurons. In the simulations presented here, the network starts with the 0.1% of the original number of neurons in each population as in the previous study and without any synaptic connections. For each population, we define a level of desired mean electrical activity based on experimental literature and a growth curve which defines the dynamics of the variation in the number of pre- and post-synaptic elements. These are again Gaussian shaped curves. Simulations were initialized without connections and only the target firing rates were defined using the original model. Job allocations for this subproject consisted on 41 nodes, deploying 1 instance of the outer loop and 8 parallel instances of this network per generation. Each instance uses 5 full nodes using MPI. Each generation was forced to end after 60 min of simulation. As in the first subproject, a penalization when the simulation did not finish on time was added to the fitness rule. Inclusion of the shape of the power spectrum in the fitness function was not performed during the time of this project, however this will be explored in future work.

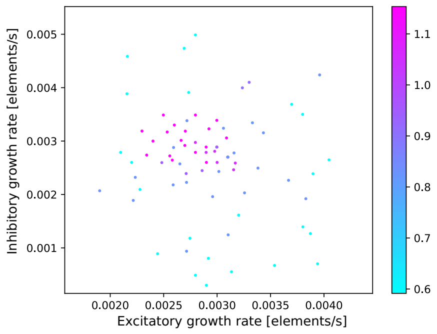


Figure 7.4: Parameter exploration for the cortical microcircuit task. Each dot in the plot represents a position in the parameter space described by the inhibitory growth rate on the y-axis and the excitatory growth rate in the x-axis. The color of the dot represents the fitness of the structural plasticity algorithm using an specific parameter combination to reach the desired activity regime. The fitness values are color coded from cyan to pink using the color bar on the right. In these tests, all excitatory populations from layers $2-3$ to 6 share the same growth rate and the same applies to all inhibitory populations.

Figure 7.4 shows the parameter exploration on 10 generations of the microcircuit model using gradient descent. The fitness of each individual is formed 80% of the inverse of the error between the average firing rate of the excitatory population and its target firing rate, and 20% of the inverse of the error between the average firing rate of the inhibitory population and its target firing rate summed up for all 4 layers. These results show that the activity requirements of this model make best use of a more balanced inhibitory and excitatory growth rates as compared to the simpler model with only two populations. This means that the best combination when using the same values for all layers is to have inhibitory synapses grow just slightly faster as excitatory synapses. The best parameter configuration depends on the flexibility we give to the model as well as the functional and structural constraints we provide to the outer loop algorithm. Given the tested setup, gradient descent provides both good exploration of the parameter space and fast convergence.

7.3 Future work

A model of the visual cortex of the macaque subdivided into 32 regions has been proposed in [135]. The stabilization of such a complexly interconnected network into desired activity regimes is not trivial, due to the influence that changes in one region can have on others, turning this into a multi-objective optimization problem. Despite being firmly grounded in experimental data, the network in its initial instantiation, exhibits bistable dynamics, with biologically unrealistic high-activity states. Schuecker et al. [135] modify the connectivity of the model systematically and, by using mean field theory, identify structural components of the network which have a strong impact on its dynamics. This methodology forms a basis for a hypothesis about essential features in the structure required to achieve realistic activity. This systematic reduction of the models dimensionality allows them to remove its bistable behaviour. However, mean field methods are severely constrained by the model complexity, and introduction of biologically plausible mechanisms, such as adaptation are difficult to include in a mean-field framework. In this subproject, we propose to use structural plasticity, guided by an optimizer, to guid such a large-scale network intro desired activity regimes.
Again, the optimizer would change connection probabilities between populations in the model. The goal would be to achieve the empirically known firing rates of the respective populations. A similar configuration has already been tested in [114] where 68 regions each composed of two populations were interconnected using DTI data for a whole human brain simulation at region scale. This work has shown that this strategy enables the identification of parameters for homeostatic rules which can reach one of multiple stable configurations of the network.

7.4 Discussion

The results of the experiments I present here elucidate how variation in the connectivity of neural networks is involved in learning and activity regulation. Structural plasticity is a phenomenon that is still in the early stages of research, particularly regarding theoretical/computational aspects. This is partially because a 'naive' implementation of such algorithms define computationally intensive processes. Additionally, large-scale biological neuronal networks are very complex systems which are difficult to characterize. If connectivity changes are taken into consideration, the complexity of the system can grow as a power of the number of nodes. With this project I have explored the potential of using both machine learning from computational science and structural plasticity inspired by neuroscience (but not necessarily physiologically realistic) to understand the sensitivity of neural networks to changes in the topology in a systematic and reproducible fashion. In addition, various structural plasticity rules found by the optimization process show desirable effects on global network properties. For example, it is possible to see that there are relationships between inhibition and excitation for every network which must be preserved in order for the network to evolve into a stable configuration. These links must still be further analyzed and linked to experimental data which can give more insight on the homeostatic regulations that the brain observes during critical periods of plasticity. Analyzing these rules will contribute to the ongoing exploration of the relevance of structural plasticity for neuronal network dynamics.

8 Conclusions, future work and discussion

8.1 Conclusions and discussion

The structural changes that neurons undergo from brain development through their lifespan is called structural plasticity. Structural plasticity plays an important role in development, healing and learning. In my thesis work I present a model of structural plasticity which can be used to modify, generate and optimize connectivity in simulations of spiking neural networks. This model has been integrated into NEST, a well established neural network simulator, in order to help the study and understanding of the relationships between structure and function in our brains. In my work I use this model as an optimization and control algorithm to fill in missing connectivity information for the simulation of neural circuits which are of interest for the neuroscientific community. I also worked in the develpment of software tools to visualize and interact with the structural plasticity algorithm which allow scientists to exploit the algorithm's potential and apply it to solve their specific questions. Besides building a software infrastructure for modeling, simulating, visualizing and analyzing plasticity in the brain, I also applied this tools to specific applications. I explored a clinical applications of the model focusing on the role of structural changes induced by Coordinated Reset Therapy for the treatment of Parkinson's disease. I also worked on another, more abstract application on a computer science, where I used structural plasticity as optimization algorithm. By using hyperparameter optimization algorithms in coordination with structural plasticity, I propose a research platform to study the effects of evolutionary tuning of learning mechanisms in the brain. In this thesis I have presented structural plasticity as a control system in neural networks which aims at optimizing its connectivity. I've shown how the implementation of a specific model of structural plasticity in NEST can be used to both, simulate progressive changes in the connections forming a neural network and to generate such connections between completely disconnected sets of neurons. With this work I provide a general overview of a framework for simulation and analysis of structural plasticity with potential applications not only in neuroscience but also for machine learning and optimization.

Current implementation: results, performance and limitations This implementation uses only random selection of targets and does not take into account the distance between neurons. The algorithm was implemented in such a way in order to enhance its performance for large scale networks. The work described in Rinke et al. [125] describes a scalable algorithm for introducing distance dependency in such a model of structural plasticity. In the future, this algorithm can be included into NEST in order to consider such constraints while maintaining a good performance at HPC scale.

Even though the current implementation of structural plasticity in NEST does not consider the weight when deciding which synapses to delete, this is compensated by other aspects of the algorithm. In this implementation, the system may create any number of connections between two neurons. By increasing the amount of connections, the strength of the contributions between both gets reinforced or reduced. This is automatically achieved as the target of reaching a given firing rate by each neuron is pursued. Even when the work in Chapter 6 shows simulations incorporating both types of plasticity, the interaction between STDP and structural plasticity is still not fully achieved. The ideal implementation would be that structural plasticity selects for deletion always the weaker synapses with a higher priority than stronger synapses. This is still work in progress and is subject of future work.

There has been already interesting findings in the computational neuroscience community using the structural plasticity implementation in NEST. The work by Gallinaro and Rotter [54] shows how this implementation is able to reproduce connectivity features observed experimentally in the visual cortex of rodents. Lu et al. [98] and Lu et al. [99] also show how experimentally measured changes in networks after stimulation can be reproduced with this framework. The role of structural plasticity in the formation of memory engrams has also been investigated and discussed in Gallinaro et al. [55] using the implementation I presented in this thesis. These examples confirm the variety of applications and new research directions enabled by the work I have presented in this thesis.

In parallel to the development of this work, collaborations have emerged with the groups developing neuromorphic hardware within the Human Brain Project. There have been discussions about a shared set of directives to describe the structural plasticity mechanisms in both software and neuromorphic hardware. Specifically, the implementation of such directives into a modeling

language such a PyNN, which can be used by BrainScales, SpiNNaker and NEST to define networks. This will help keep a common language among simulators/emulators which can be used later as a base to compare the different implementations of structural plasticity. I co-organized several workshops [119] in order to discuss requirements, challenges and applications together with developers and users around the topic of structural plasticity. This allowed me to have continuous feedback on the framework usability but also interactions and developments in other research groups. Examples of these parallel developments are the work from Bogdan et al. [15] and Schemmel et al. [132] on neuromorphic hardware.
The work in Bogdan et al. [15] and Bogdan and Furber [14] describes the implementation of structural plasticity in SpiNNaker. It includes distance dependency in the selection of targets and has a limit in the total number of connections which can be made per neuron. The restrictions imposed by the neuromorphic chip limit the total number of neurons which can be simulated. However, under these parameters, the performance is very high, allowing almost real time simulations. This implementation has been applied to motion recognition and learning of rules to win simple games like pong.
An model of structural plasticity in BrainScales Schemmel et al. [132] has also been successfully implemented and used for machine learning tasks. BrainScales is an analog emulation system with digital interfaces. The analog nature of such a system induces some inhomogeneities in each chip which have to be taken into consideration during emulations of a network. An interesting application of structural plasticity in this architecture is that they have used it to automatically learn which connections are physically more suitable for a given network model [12]. This is a unique usage of the structural plasticity concept which links to the analog nature of human brains as well.

Visualization and steering Visualization is an essential tool to understand how these complex simulations of neural networks behave. The dynamics in the connectivity which structural plasticity introduces, make it even more important to observe the network in order to understand its configuration and behaviour. In the end, the goal is to be able to relate how changes in the connectivity affect the activity in the network. If we managed to elucidate these relationships, we would be able to better understand why our brains are wired as they are and, hopefully, how higher function emerges as well.
The brain is a very robust system, resilient to errors, unknown input, damage, and stressing situations. It seems reasonable to think that in order to achieve such a high level of resilience, the brain

has a redundant configuration scheme which is prepared for failures in single circuits. For this reason, several connectivity configurations might in the end provide similar output under a single measure of performance. In this thesis I have presented an interactive steering and visualization framework which assist the neuroscientists in finding and understanding connectivity configuration in their models. With this system it is possible to navigate the connectivity configurations and select the ones which are biologically and functionally meaningful for answering a particular scientific question. In combination with the MSPViz tool, the scientists can also understand and analyze with even more detail the changes in the connectivity performed by the structural plasticity algorithm.

Clinical applications In this thesis I have also shown an application of the structural plasticity algorithm to a clinical problem. The work described in Chapter 6 provides to the scientific community a simulation framework to study the impact of slow structural changes in a therapeutic setup. By including structural and synaptic plasticity in this model, the user can observe and analyze the impact of deep brain stimulation in a realistic model of the Sub Thalamic Nucleus and the Globus Plidus externus. In this work I have found that adding structural plasticity as a homeostatic controller in our simulations allows us to reproduce the long lasting effects of Coordinated Reset Therapy. This supports our initial theory that waiting times between therapeutic stimulation are to be considered part of the therapy and should be well defined in order to optimize its benefits. The way I have combined short and long time scale plasticity effects into NEST simulations allows to keep computational performance and biological meaningfulness.

Learning and optimization The last part of this thesis is related to learning and optimization of network connectivity to reach particular dynamical features of the network. Using the learning to learn paradigm, we have optimized the hyperparameters of the structural plasticity algorithm in order to achieve two goals, enhanced correlations and optimized connectivity. In the future, this work can be applied to more defined tasks such as pattern recognition and information coding, among others. Learning to learn allows us to unveil the not so evident links between the homeostatic rules and the end dynamics of the system. The more complex the target measure for structural plasticity is, the more difficult it is to create a direct rule between connectivity changes (control) and the desired output (error reduction).

Final remarks In the introduction, I listed some essential features which define structural plasticity. The algorithm that I have implemented, used, analyzed, optimized and discussed in this thesis is not complete. The algorithm already incorporates most of the elements in the aforementioned list, however, there is room for improvement in the performance and ways in which the structural changes take place. For example, the distance dependency factor in the selection of targets is still to be incorporated and defined. The performance of the algorithm can also be enhanced by introducing simplifications in the communication of connectivity data. Even though the applications and the implementation showcased here are focused on point neurons, all the concepts developed here can be applied also to morphologically detailed neurons. This can be done by the integration of models for axonal and dendritic growth and spatial distribution of synaptic contacts along the morphology of the neuron.
This thesis was carried out in the context of the Human Brain Project and develops on a topic which has been so far not sufficiently addressed by the computational neuroscience community. In this same context, this thesis work has been a point of interaction with similar developments in simulations [15] and emulations [132] on neuromorphic hardware. It is currently a topic of discussion if a stand alone simulator of structural plasticity, compatible with different simulators and even with neuromorphic hardware, is a good way to implement future developments. A standard definition of structural plasticity in PyNN, as mentioned in [15], is also considered as a desirable next step in order to ease the usability and adoption of this model.
Adding dynamic connectivity to a neural network simulation is not straight forward. It impacts the performance of the simulations as well as the dynamics of the system and our ability to use well established analytical tools to understand the behaviour of the network. From the point in time when the brain starts to develop, the building of the connections between neurons has a well defined set of rules which are still not completely understood today. The mature brain also changes in a consistent way. It is undeniable that structural changes take place in the brain during the whole lifespan and that they are part of critical functions such as learning and healing. With this thesis work I have addressed both the implementation and analysis of a model of structural plasticity which can be used in large scale neural networks. I have also discussed potential applications of such an algorithm and defined its current limitations. The work exposed here sets ground and motivates future implementations of such an algorithm on software and neuromorphic hardware. With these tools is now possible to better understand the dynamics of connectivity in neural networks and the links between structure and function in the brain.

List of Figures

List of Tables

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] Greg Abram and Lloyd Treinish. An extended data-flow architecture for data analysis and visualization. In *Proceedings of the 6th Conference on Visualization'95*, page 263, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7187-4.

[4] N. A. Akar, B. Cumming, V. Karakasis, A. Küsters, W. Klijn, A. Peyser, and S. Yates. Arbor — A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures. In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 274–282, feb 2019. doi: 10.1109/EMPDP.2019.8671560.

[5] F. A. al Mohanna, J. Cave, and S. R. Bolsover. A narrow window of intracellular calcium concentration is optimal for neurite outgrowth in rat sensory neurones. *Brain Res. Dev. Brain Res.*, 70(2):287–290, Dec 1992.

[6] MM Ali and P Kaelo. Improved particle swarm algorithms for global optimization. *Applied mathematics and computation*, 196(2):578–593, 2008.

[7] Katrin Amunts, Christoph Ebell, Jeff Muller, Martin Telefont, Alois Knoll, and Thomas Lippert. The human brain project: creating a european research infrastructure to decode the human brain. *Neuron*, 92(3):574–581, 2016.

[8] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. pages 3981–3989, 2016.

[9] Jyotika Bahuguna, Tom Tetzlaff, Arvind Kumar, Jeanette Hellgren Kotaleski, and Abigail Morrison. Homologous basal ganglia network models in physiological and parkinsonian conditions. *Frontiers in computational neuroscience*, 11:79, 2017.

[10] Rembrandt Bakker, Thomas Wachtler, and Markus Diesmann. Cocomac 2.0 and the future of tract-tracing databases. *Frontiers in Neuroinformatics*, 6:30, 2012.

[11] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017.

[12] Sebastian Billaudelle, Benjamin Cramer, Mihai A. Petrovici, Korbinian Schreiber, David Kappel, Johannes Schemmel, and Karlheinz Meier. Structural plasticity on an accelerated analog neuromorphic hardware system, 2020.

[13] Tim Blackwell and Jürgen Branke. Multi-swarm optimization in dynamic environments. In *Workshops on Applications of Evolutionary Computation*, pages 489–500. Springer, 2004.

[14] Petrut A Bogdan and Stephen Furber. Neurogenesis on spinnaker. In *1st Human Brain Project Student Conference*, page 75.

[15] Petruț Antoniu Bogdan, Andrew Graham David Rowley, Oliver Rhodes, and Steve B Furber. Structural plasticity on the spinnaker many-core neuromorphic system. *Frontiers in Neuroscience*, 12:434, 2018.

[16] Hannah Bos, Universitätsprofessor Dr Björn Kampa, and Gaute T Einevoll. Connectivity structure induced dynamics and correlations in spiking neural networks.

[17] Hannah Bos, Abigail Morrison, Alexander Peyser, Jan Hahne, Moritz Helias, Susanne Kunkel, Tammo Ippen, Jochen Martin Eppler, Maximilian Schmidt, Alex Seeholzer, Mikael Djurfeldt, Sandra Diaz, Janne Morén, Rajalekshmi Deepu, Teo Stocco, Moritz Deger, Frank Michler, and Hans Ekkehard Plesser. NEST 2.10.0, December 2015. URL `http://dx.doi.org/10.5281/zenodo.44222`.

[18] Hannah Bos, Markus Diesmann, and Moritz Helias. Identifying anatomical origins of coexisting oscillations in the cortical microcircuit. *PLoS computational biology*, 12(10): e1005132, 2016.

[19] Nadia Boukhelifa and Peter J Rodgers. A model and software system for coordinated and multiple views in exploratory visualization. *Information Visualization*, 2(4):258–269, 2003. ISSN 1473-8716. doi: 10.1057/palgrave.ivs.9500057.

[20] Janina Boyke, Joenna Driemeyer, Christian Gaser, Christian Büchel, and Arne May. Training-induced brain structure changes in the elderly. *Journal of Neuroscience*, 28(28): 7031–7035, 2008.

[21] Juan Pedro Brito Mendéz, Juan Bautista Hernándo Vieites, Benjamin Weyers, and Vicente Martín Ayuso. Visualization of large distributed data in neuroscience. 2016.

[22] M. Butz and A. van Ooyen. A simple rule for dendritic spine and axonal bouton formation can account for cortical reorganization after focal retinal lesions. *PLoS Comput. Biol.*, 9 (10):e1003259, Oct 2013.

[23] M. Butz, I. D. Steenbuck, and A. van Ooyen. Homeostatic structural plasticity increases the efficiency of small-world networks. *Front Synaptic Neurosci*, 6:7, 2014.

[24] Markus Butz and Arjen van Ooyen. A simple rule for dendritic spine and axonal bouton formation can account for cortical reorganization after focal retinal lesions. *PLoS Comput Biol*, 9(10):e1003259, 2013.

[25] Nicholas T Carnevale and Michael L Hines. *The NEURON book*. Cambridge University Press, 2006.

[26] Augustin Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

[27] Hank Childs, Eric S. Brugger, Kathleen S. Bonnell, Jeremy S. Meredith, Mark Miller, Brad J. Whitlock, and Nelson Max. A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005*, pages 190–198, 2005.

[28] Robert J Coffey. Deep brain stimulation devices: a brief technical history and review. *Artificial organs*, 33(3):208–220, 2009.

[29] Jack D Cowan. Stochastic neurodynamics. In *Advances in neural information processing systems*, pages 62–69, 1991.

[30] Toby S Cubitt, Jens Eisert, and Michael M Wolf. Extracting dynamical equations from experimental data is np hard. *Physical review letters*, 108(12):120503, 2012.

[31] JL Dantzker and EM Callaway. Laminar sources of synaptic input to cortical inhibitory interneurons and pyramidal neurons. *Nature neuroscience*, 3(7):701, 2000.

[32] Hugo De Garis, Chen Shuo, Ben Goertzel, and Lian Ruiting. A world survey of artificial brain projects, part i: Large-scale brain simulations. *Neurocomputing*, 74(1):3–29, 2010.

[33] V. De Paola, A. Holtmaat, G. Knott, S. Song, L. Wilbrecht, P. Caroni, and K. Svoboda. Cell type-specific structural plasticity of axonal branches and boutons in the adult neocortex. *Neuron*, 49(6):861–875, Mar 2006.

[34] Gustavo Deco, Viktor K. Jirsa, Peter A. Robinson, Michael Breakspear, and Karl Friston. The dynamic brain: From spiking neurons to neural masses and cortical fields. *PLoS Comput Biol*, 4(8):e1000092, 08 2008.

[35] Gustavo Deco, Adrián Ponce-Alvarez, Dante Mantini, Gian Luca Romani, Patric Hagmann, and Maurizio Corbetta. Resting-state functional connectivity emerges from structurally and dynamically shaped slow linear fluctuations. *The Journal of Neuroscience*, 33(27): 11239–11252, 2013.

[36] Gustavo Deco, Adrián Ponce-Alvarez, Patric Hagmann, Gian Luca Romani, Dante Mantini, and Maurizio Corbetta. How local excitation–inhibition ratio impacts the whole brain dynamics. *The Journal of Neuroscience*, 34(23):7886–7898, 2014.

[37] Sandra Diaz-Pier, Mikaël Naveau, Markus Butz-Ostendorf, and Abigail Morrison. Automatic generation of connectivity for large-scale neuronal network models through structural

plasticity. *Frontiers in Neuroanatomy*, 10(57), 2016. ISSN 1662-5129. doi: 10.3389/fnana.2016.00057. URL `http://www.frontiersin.org/neuroanatomy/10.3389/fnana.2016.00057/abstract`.

[38] Mikael Djurfeldt, Johannes Hjorth, Jochen M. Eppler, Niraj Dudani, Moritz Helias, Tobias C. Potjans, Upinder S. Bhalla, Markus Diesmann, Jeanette Hellgren Kotaleski, and Örjan Ekeberg. Run-time interoperability between neuronal network simulators based on the music framework. *Neuroinformatics*, 8(1):43–60, 2010. ISSN 1559-0089. doi: 10.1007/s12021-010-9064-z. URL `http://dx.doi.org/10.1007/s12021-010-9064-z`.

[39] Marco Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[40] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[41] Julien Doyon and Habib Benali. Reorganization and plasticity in the adult brain during learning of motor skills. *Current opinion in neurobiology*, 15(2):161–167, 2005.

[42] Salvador Dura-Bernal, Benjamin A Suter, Padraig Gleeson, Matteo Cantarelli, Adrian Quintana, Facundo Rodriguez, David J Kedziora, George L Chadderdon, Cliff C Kerr, Samuel A Neymotin, et al. Netpyne, a tool for data-driven multiscale modeling of brain circuits. *Elife*, 8:e44494, 2019.

[43] Egidio D'Angelo, Sergio Solinas, Jesus Garrido, Claudia Casellato, Alessandra Pedrocchi, Jonathan Mapelli, Daniela Gandolfi, and Francesca Prestori. Realistic modeling of neurons and networks: towards brain simulation. *Functional neurology*, 28(3):153, 2013.

[44] Russell C Eberhart, Doyle J Groves, and Joshua K Woodward. Deep swarm: Nested particle swarm optimization. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–6. IEEE, 2017.

[45] Martin Ebert, Christian Hauptmann, and Peter A Tass. Coordinated reset stimulation in a large-scale model of the stn-gpe circuit. *Frontiers in computational neuroscience*, 8:154, 2014.

[46] Gaute T Einevoll, Alain Destexhe, Markus Diesmann, Sonja Grün, Viktor Jirsa, Marc de Kamps, Michele Migliore, Torbjørn V Ness, Hans E Plesser, and Felix Schürmann. The scientific case for brain simulations. *Neuron*, 102(4):735–744, 2019.

[47] Chris Eliasmith and Oliver Trujillo. The use and abuse of large-scale brain models. *Current Opinion in Neurobiology*, 25:1 – 6, 2014.

[48] Jochen M Eppler, Moritz Helias, Eilif Muller, Markus Diesmann, and Marc-Oliver Gewaltig. PyNEST: a convenient interface to the NEST simulator. *Frontiers in Neuroinformatics*, 2(12), 2009. ISSN 1662-5196. doi: 10.3389/neuro.11.012.2008.

[49] Jochen Martin Eppler, Robin Pauli, Alexander Peyser, Tammo Ippen, Abigail Morrison, Johanna Senk, Wolfram Schenck, Hannah Bos, Moritz Helias, Maximilian Schmidt, Susanne Kunkel, Jakob Jordan, Marc-Oliver Gewaltig, Claudia Bachmann, Jannis Schuecker, Sacha Albada, Tiziano Zito, Moritz Deger, Frank Michler, Espen Hagen, Hesam Setareh, Luis Riquelme, Ali Shirvani, Renato Duarte, Rajalekshmi Deepu, and Hans Ekkehard Plesser. Nest 2.8.0, September 2015. URL `http://dx.doi.org/10.5281/zenodo.32969`.

[50] Kirk I Erickson, Michelle W Voss, Ruchika Shaurya Prakash, Chandramallika Basak, Amanda Szabo, Laura Chaddock, Jennifer S Kim, Susie Heo, Heloisa Alves, Siobhan M White, et al. Exercise training increases size of hippocampus and improves memory. *Proceedings of the National Academy of Sciences*, 108(7):3017–3022, 2011.

[51] D.C. Van Essen and K. Ugurbil. The future of the human connectome. *NeuroImage*, 62(2): 1299 – 1310, 2012.

[52] D.C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T.E.J. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S.W. Curtiss, S. Della Penna, D. Feinberg, M.F. Glasser, N. Harel, A.C. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S.E. Petersen, F. Prior, B.L. Schlaggar, S.M. Smith, A.Z. Snyder, J. Xu, and E. Yacoub. The human connectome project: A data acquisition perspective. *NeuroImage*, 62(4):2222 – 2231, 2012.

[53] Nathan Fabian, Kenneth Moreland, David Thompson, Andrew C. Bauer, Patrick Marion, Berk Geveci, Michel E. Rasquin, and Kenneth E. Jansen. The paraview coprocessing library: A scalable, general purpose in situ visualization library. In David

Rogers and Cláudio T. Silva, editors, *LDAV*, pages 89–96. IEEE, 2011. ISBN 978-1-4673-0156-5. URL `http://dblp.uni-trier.de/db/conf/ldav/ldav2011.html#FabianMTBMGRJ11`.

[54] Júlia V Gallinaro and Stefan Rotter. Associative properties of structural plasticity based on firing rate homeostasis in recurrent neuronal networks. *Scientific reports*, 8(1):3754, 2018.

[55] Julia V Gallinaro, Nebojsa Gasparovic, and Stefan Rotter. Homeostatic structural plasticity leads to the formation of memory engrams through synaptic rewiring in recurrent networks. *bioRxiv*, 2020.

[56] Lynne V Gauthier, Edward Taub, Christi Perkins, Magdalene Ortmann, Victor W Mark, and Gitendra Uswatte. Remodeling the brain - plastic structural brain changes produced by different motor therapies after stroke. *Stroke*, 39(5):1520–1525, 2008.

[57] Kimberly Gerrow and Craig E Brown. Structural neural plasticity during stroke recovery. In *The Rewiring Brain*, pages 49–70. Elsevier, 2017.

[58] M. O. Gewaltig and M Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2(4): 1440, 2007.

[59] Marc-Oliver Gewaltig and Markus Diesmann. NEST (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[60] Edward George Gray. Electron microscopy of collagen-like connective tissue fibrils of an insect. *Proc. R. Soc. Lond. B*, 150(939):233–239, 1959.

[61] Daniel Harnack, Miha Pelko, Antoine Chaillet, Yacine Chitour, and Mark CW van Rossum. Stability of neuronal networks with homeostatic regulation. *PLoS computational biology*, 11(7):e1004357, 2015.

[62] Christian Hauptmann and Peter A Tass. Therapeutic rewiring by means of desynchronizing brain stimulation. *Biosystems*, 89(1-3):173–181, 2007.

[63] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[64] Moritz Helias, Susanne Kunkel, Gen Masumoto, Jun Igarashi, Jochen Martin Eppler, Shin Ishii, Tomoki Fukai, Abigail Morrison, and Markus Diesmann. Supercomputers ready for use as discovery machines for neuroscience. *Frontiers in neuroinformatics*, 6:26, 2012.

[65] Amy Henderson. *The ParaView Guide: A Parallel Visualization Application*. Kitware, November 2004.

[66] Keith B Hengen, Mary E Lambo, Stephen D Van Hooser, Donald B Katz, and Gina G Turrigiano. Firing rate homeostasis in visual cortex of freely behaving rodents. *Neuron*, 80(2):335–342, 2013.

[67] Keith B Hengen, Alejandro Torrado Pacheco, James N McGregor, Stephen D Van Hooser, and Gina G Turrigiano. Neuronal firing rate homeostasis is inhibited by sleep and promoted by wake. *Cell*, 165(1):180–191, 2016.

[68] T. K. Hensch. Critical period plasticity in local cortical circuits. *Nat. Rev. Neurosci.*, 6(11): 877–888, Nov 2005.

[69] Takao K Hensch. Critical period plasticity in local cortical circuits. *Nature Reviews Neuroscience*, 6(11):877–888, 2005.

[70] Iain Hepburn, Weiliang Chen, Stefan Wils, and Erik De Schutter. Steps: efficient simulation of stochastic reaction–diffusion models in realistic morphologies. *BMC systems biology*, 6 (1):1–19, 2012.

[71] Berk Hess, Carsten Kutzner, David Van Der Spoel, and Erik Lindahl. Gromacs 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of chemical theory and computation*, 4(3):435–447, 2008.

[72] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[73] H Holland John. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. *USA: University of Michigan*, 1975.

[74] A. Holtmaat and K. Svoboda. Experience-dependent structural synaptic plasticity in the mammalian brain. *Nat. Rev. Neurosci.*, 10(9):647–658, Sep 2009.

[75] Anthony J.G.D. Holtmaat, Joshua T. Trachtenberg, Linda Wilbrecht, Gordon M. Shepherd, Xiaoqun Zhang, Graham W. Knott, and Karel Svoboda. Transient and persistent dendritic spines in the neocortex in vivo. *Neuron*, 45(2):279 – 291, 2005. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2005.01.003. URL `http://www.sciencedirect.com/science/article/pii/S0896627305000048`.

[76] David H Hubel and Torsten N Wiesel. The period of susceptibility to the physiological effects of unilateral eye closure in kittens. *The Journal of physiology*, 206(2):419–436, 1970.

[77] Jakob Jordan, Tammo Ippen, Moritz Helias, Itaru Kitayama, Mitsuhisa Sato, Jun Igarashi, Markus Diesmann, and Susanne Kunkel. Extremely scalable spiking neuronal network simulation code: From laptops to exascale computers. *Frontiers in Neuroinformatics*, 12:2, 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00002. URL `https://www.frontiersin.org/article/10.3389/fninf.2018.00002`.

[78] Jakob Jordan, Tammo Ippen, Moritz Helias, Itaru Kitayama, Mitsuhisa Sato, Jun Igarashi, Markus Diesmann, and Susanne Kunkel. Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Frontiers in neuroinformatics*, 12:2, 2018.

[79] Abhaya Chandra Kammara, Lingaselvan Palanichamy, and Andreas König. Multi-objective optimization and visualization for analog design automation. *Complex & Intelligent Systems*, 2(4):251–267, 2016.

[80] David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Network plasticity as bayesian inference. *PLoS computational biology*, 11(11):e1004485, 2015.

[81] David Kappel, Robert Legenstein, Stefan Habenschuss, Michael Hsieh, and Wolfgang Maass. Reward-based stochastic self-configuration of neural circuits. *arXiv preprint arXiv:1704.04238*, 2017.

[82] T. Keck, T. D. Mrsic-Flogel, M. Vaz Afonso, U. T. Eysel, T. Bonhoeffer, and M. Hubener.

Massive restructuring of neuronal circuits during functional reorganization of adult visual cortex. *Nat. Neurosci.*, 11(10):1162–1167, Oct 2008.

[83] T. Keck, V. Scheuss, R. I. Jacobsen, C. J. Wierenga, U. T. Eysel, T. Bonhoeffer, and M. Hubener. Loss of sensory input causes rapid structural changes of inhibitory neurons in adult mouse visual cortex. *Neuron*, 71(5):869–882, Sep 2011.

[84] T. Keck, G. B. Keller, R. I. Jacobsen, U. T. Eysel, T. Bonhoeffer, and M. Hubener. Synaptic scaling and homeostatic plasticity in the mouse visual cortex in vivo. *Neuron*, 80(2): 327–334, Oct 2013.

[85] Vassilis Kehayas and Anthony Holtmaat. Structural plasticity and cortical connectivity. In *The Rewiring Brain*, pages 3–26. Elsevier, 2017.

[86] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[87] Melinda S Kelley and Oswald Steward. Injury-induced physiological events that may modulate gene expression in neurons and glia. *Reviews in the Neurosciences*, 8(3-4): 147–178, 1997.

[88] James Kennedy and Russell C Eberhart. The particle swarm: social adaptation in information-processing systems. In *New ideas in optimization*, pages 379–388. McGraw-Hill Ltd., UK, 1999.

[89] D.E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering. Dover Publications, 2012. ISBN 9780486135076. URL `https://books.google.de/books?id=onuH0PnZwV4C`.

[90] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[91] S. A. Kirov, C. A. Goddard, and K. M. Harris. Age-dependence in the homeostatic upregulation of hippocampal dendritic spine number during blocked synaptic transmission. *Neuropharmacology*, 47(5):640–648, Oct 2004.

[92] Jeffrey A Kleim and Theresa A Jones. Principles of experience-dependent neural plasticity: implications for rehabilitation after brain damage. *Journal of speech, language, and hearing research*, 2008.

[93] Susanne Kunkel, Maximilian Schmidt, Jochen M Eppler, Hans E Plesser, Gen Masumoto, Jun Igarashi, Shin Ishii, Tomoki Fukai, Abigail Morrison, Markus Diesmann, et al. Spiking network simulation code for petascale computers. *Frontiers in neuroinformatics*, 8, 2014.

[94] Susanne Kunkel, Maximilian Schmidt, Jochen Martin Eppler, Hans Ekkehard Plesser, Gen Masumoto, Jun Igarashi, Shin Ishii, Tomoki Fukai, Abigail Morrison, Markus Diesmann, and Moritz Helias. Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics*, 8(78), 2014. ISSN 1662-5196. doi: 10.3389/fninf.2014.00078. URL `http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00078/abstract`.

[95] Thomas Lippert and Boris Orth. *Supercomputing Infrastructure for Simulations of the Human Brain*, pages 198–212. Springer International Publishing, Cham, 2014. ISBN 978-3-319-12084-3. doi: 10.1007/978-3-319-12084-3_16. URL `http://dx.doi.org/10.1007/978-3-319-12084-3_16`.

[96] S. A. Lipton and S. B. Kater. Neurotransmitter regulation of neuronal outgrowth, plasticity and survival. *Trends Neurosci.*, 12(7):265–270, Jul 1989.

[97] Armando López-Cuevas, Bernardino Castillo-Toledo, Laura Medina-Ceja, and Consuelo Ventura-Mejía. State and parameter estimation of a neural mass model from electrophysiological signals during the status epilepticus. *NeuroImage*, 113:374–386, 2015.

[98] Han Lu, Júlia V. Gallinaro, and Stefan Rotter. Network remodeling induced by transcranial brain stimulation: A computational model of tdcs-triggered cell assembly formation. *Network Neuroscience*, 0(0):1–21, 0. doi: 10.1162/netn_a_00097. URL `https://doi.org/10.1162/netn_a_00097`.

[99] Han Lu, Julia V Gallinaro, Claus Normann, Stefan Rotter, and Ipek Yalcin. Time course of homeostatic structural plasticity in response to optogenetic stimulation in mouse anterior cingulate cortex. *bioRxiv*, 2020.

[100] Sebastian Lührs. Jube-a flexible, application-and platform-independent environment for benchmarking. In *4th JLESC Workshop*, number FZJ-2015-07252. Jülich Supercomputing Center, 2015.

[101] Borys Lysyansky, Oleksandr V Popovych, and Peter A Tass. Desynchronizing anti-resonance effect of m: n on–off coordinated reset stimulation. *Journal of Neural Engineering*, 8(3):036019, 2011.

[102] Arianna Maffei and Gina G Turrigiano. Multiple modes of network homeostasis in visual cortical layer 2/3. *Journal of Neuroscience*, 28(17):4377–4384, 2008.

[103] Thanos Manos, Sandra Diaz-Pier, and Peter Tass. 'long-term desynchronization by coordinated reset stimulation in a neural network model with synaptic and structural plasticity. *PLOS Computational Biology - Under revision*.

[104] Kresimir Matkovic, Denis Gracanin, Mario Jelovic, and Helwig Hauser. Interactive visual steering-rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, 2008.

[105] Kresimir Matkovic, Denis Gracanin, Rainer Splechtna, Mario Jelovic, Benedikt Stehno, Helwig Hauser, and Werner Purgathofer. Visual analytics for complex engineering systems: Hybrid visual steering of simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1803–1812, 2014.

[106] Arne May. Experience-dependent structural plasticity in the adult human brain. *Trends in cognitive sciences*, 15(10):475–482, 2011.

[107] Brenda M Michelson. Event-driven architecture overview. *Patricia Seybold Group*, 2, 2006. doi: 10.1571/bda2-2-06cc.

[108] Michele Migliore, Francesco Cavarretta, Michael L. Hines, and Gordon M. Shepherd. Distributed organization of a brain microcircuit analyzed by three-dimensional modeling: the olfactory bulb. *Frontiers in Computational Neuroscience*, 8:50, 2014. ISSN 1662-5188. doi: 10.3389/fncom.2014.00050. URL `http://journal.frontiersin.org/article/10.3389/fncom.2014.00050`.

[109] Tom M Mitchell. Machine learning and data mining. *Communications of the ACM*, 42 (11):30–36, 1999.

[110] Chris North and Ben Shneiderman. A taxonomy of multiple window coordination, 1997. URL `http://drum.lib.umd.edu//handle/1903/5892`.

[111] Chris North and Ben Shneiderman. Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pages 128–135, New York, NY, USA, 2000. ACM. ISBN 1-58113-252-2. doi: 10.1145/345513.345282. URL `http://doi.acm.org/10.1145/345513.345282`.

[112] Christian Nowke, Daniel Zielasko, Benjamin Weyers, Bernd Hentschel, Alexander Peyser, and Torsten Kuhlen. Integrating visualizations into modeling NEST simulations. *Frontiers in Neuroinformatics*, 9(29), 2015. ISSN 1662-5196. doi: 10.3389/fninf.2015.00029. URL `http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2015.00029/abstract`.

[113] Christian Nowke, Daniel Zielasko, Benjamin Weyers, Alexander Peyser, Bernd Hentschel, and Torsten W Kuhlen. Integrating visualizations into modeling nest simulations. *Frontiers in neuroinformatics*, 9:29, 2015.

[114] Christian Nowke, Sandra Diaz-Pier, Benjamin Weyers, Bernd Hentschel, Abigail Morrison, Torsten W Kuhlen, and Alexander Peyser. Toward rigorous parameterization of underconstrained neural network models through interactive visualization and steering of connectivity generation. *Frontiers in Neuroinformatics*, 12, 2018.

[115] Toru Ohira and Jack D Cowan. Master-equation approach to stochastic neurodynamics. *Physical Review E*, 48(3):2259, 1993.

[116] H. J. Park and K. Friston. Structural and functional brain networks: from connections to cognition. *Science*, 342(6158):1238411, Nov 2013.

[117] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[118] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[119] Alexander Peyser, Sandra Diaz, and Wouter Klijn. Workshop on generative connectomics and plasticity. CNS*2019 Workshop, Barcelona (Spain), 16 Jul 2019 - 17 Jul 2019, Jul 2019. URL https://juser.fz-juelich.de/record/865118.

[120] Hans E Plesser, Jochen M Eppler, Abigail Morrison, Markus Diesmann, and Marc-Oliver Gewaltig. Efficient parallel simulation of large-scale neuronal networks on clusters of multiprocessor computers. In *Euro-Par 2007 parallel processing*, pages 672–681. Springer, 2007.

[121] Oleksandr V Popovych, Markos N Xenakis, and Peter A Tass. The spacing principle for unlearning abnormal neuronal synchrony. *PloS one*, 10(2):e0117205, 2015.

[122] T. C. Potjans and M. Diesmann. The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model. *Cereb. Cortex*, 24(3):785–806, Mar 2014.

[123] Julia Reckfort, Hendrik Wiese, Melanie Dohmen, David Grässel, Uwe Pietrzyk, Karl Zilles, Katrin Amunts, and Markus Axer. Extracting the inclination angle of nerve fibers within the human brain with 3d-pli independent of system properties. *Proc. SPIE*, 8873: 88730F–88730F–10, 2013. doi: 10.1117/12.2023198. URL http://dx.doi.org/10.1117/12.2023198.

[124] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21, pages 25–34. ACM, 1987.

[125] Sebastian Rinke, Mikaël Naveau, Felix Wolf, and Markus Butz-Ostendorf. Critical periods emerge from homeostatic structural plasticity in a full-scale model of the developing cortical column. In *The Rewiring Brain*, pages 177–201. Elsevier, 2017.

[126] Jonathan C. Roberts. State of the art: coordinated & multiple views in exploratory visualization. *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*, (Cmv):61–71, July 2007. doi: 10.1109/CMV.2007.20. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4269947`.

[127] Dipanjan Roy, Rodrigo Sigala, Michael Breakspear, Anthony Randal McIntosh, Viktor K Jirsa, Gustavo Deco, and Petra Ritter. Using the virtual brain to reveal the role of oscillations and plasticity in shaping brain's dynamical landscape. *Brain connectivity*, 4(10): 791–811, 2014.

[128] Jonathan E Rubin and David Terman. High frequency stimulation of the subthalamic nucleus eliminates pathological thalamic rhythmicity in a computational model. *Journal of computational neuroscience*, 16(3):211–235, 2004.

[129] Y. S. Ryu, B. Yost, G. Convertino, J. Chen, and C. North. Exploring cognitive strategies for integrating multiple-view visualizations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 47(3):591–595, October 2003. ISSN 1071-1813. doi: 10.1177/154193120304700371. URL `http://pro.sagepub.com/lookup/doi/10.1177/154193120304700371`.

[130] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[131] Paula Sanz Leon, Stuart Knock, M. Woodman, Lia Domide, Jochen Mersmann, Anthony R McIntosh, and Viktor Jirsa. The virtual brain: a simulator of primate brain network dynamics. *Frontiers in Neuroinformatics*, 7:10, 2013. ISSN 1662-5196. doi: 10.3389/fninf.2013.00010. URL `http://journal.frontiersin.org/article/10.3389/fninf.2013.00010`.

[132] Johannes Schemmel, Laura Kriener, Paul Müller, and Karlheinz Meier. An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based non-linear dendrites. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2217–2226. IEEE, 2017.

[133] Michael Schirner, Anthony Randal McIntosh, Viktor Jirsa, Gustavo Deco, and Petra Ritter. Bridging multiple scales in the human brain using computational modelling. *bioRxiv*, 2016.

doi: 10.1101/085548. URL http://biorxiv.org/content/early/2016/11/03/085548.

[134] Jannis Schuecker, Maximilian Schmidt, Sacha J van Albada, Markus Diesmann, and Moritz Helias. Fundamental activity constraints lead to specific interpretations of the connectome. *arXiv preprint arXiv:1509.03162*, 2015.

[135] Jannis Schuecker, Maximilian Schmidt, Sacha J van Albada, Markus Diesmann, and Moritz Helias. Fundamental activity constraints lead to specific interpretations of the connectome. *PLoS computational biology*, 13(2):e1005179, 2017.

[136] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[137] Bernhard Sehm, Marco Taubert, Virginia Conde, David Weise, Joseph Classen, Juergen Dukart, Bogdan Draganski, Arno Villringer, and Patrick Ragert. Structural brain plasticity in parkinson's disease induced by balance training. *Neurobiology of aging*, 35(1):232–239, 2014.

[138] Goran Söhl, Stephan Maxeiner, and Klaus Willecke. Expression and functions of neuronal gap junctions. *Nature reviews neuroscience*, 6(3):191, 2005.

[139] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS Comput Biol*, 1(4):e42, 2005.

[140] Marcel Stimberg, Romain Brette, and Dan FM Goodman. Brian 2, an intuitive and efficient neural simulator. *eLife*, 8:e47314, August 2019. ISSN 2050-084X. doi: 10.7554/eLife.47314.

[141] R Paul Stroemer, Thomas A Kent, and Claire E Hulsebosch. Enhanced neocortical neural sprouting, synaptogenesis, and behavioral recovery with d-amphetamine therapy after neocortical infarction in rats. *Stroke*, 29(11):2381–2395, 1998.

[142] Anand Subramoney, Sandra Diaz-Pier, Arjun Rao, Franz Scherr, Darjan Salaj, Thomas Bohnstingl, Jakob Jordan, Nikolaus Kopp, Daniel Hackhofer, and Sinisa Stekovic. Igitugraz/l2l: v1.0.0-beta, March 2019. URL https://doi.org/10.5281/zenodo.2590760.

[143] Riccardo Taormina and Kwok-Wing Chau. Data-driven input variable selection for rainfall–runoff modeling using binary-coded particle swarm optimization and extreme learning machines. *Journal of Hydrology*, 529:1617–1632, 2015.

[144] Peter A Tass. Desynchronization by means of a coordinated reset of neural subpopulationsa novel technique for demand-controlled deep brain stimulation. *Progress of Theoretical Physics Supplement*, 150:281–296, 2003.

[145] Peter A Tass. A model of desynchronizing deep brain stimulation with a demand-controlled coordinated reset of neural subpopulations. *Biological cybernetics*, 89(2):81–88, 2003.

[146] Peter A Tass, Li Qin, Christian Hauptmann, Sandra Dovero, Erwan Bezard, Thomas Boraud, and Wassilios G Meissner. Coordinated reset has sustained aftereffects in parkinsonian monkeys. *Annals of neurology*, 72(5):816–820, 2012.

[147] David Terman, Jonathan E Rubin, AC Yew, and CJ Wilson. Activity patterns in a model for the subthalamopallidal network of the basal ganglia. *Journal of Neuroscience*, 22(7): 2963–2976, 2002.

[148] Cibu Thomas and Chris I Baker. Teaching an adult brain new tricks: a critical review of evidence for training-dependent structural plasticity in humans. *NeuroImage*, 73:225–236, 2013.

[149] Alex M Thomson, David C West, Yun Wang, and A Peter Bannister. Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2–5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral cortex*, 12(9):936–953, 2002.

[150] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[151] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[152] J. T. Trachtenberg, B. E. Chen, G. W. Knott, G. Feng, J. R. Sanes, E. Welker, and K. Svoboda. Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature*, 420(6917):788–794, 2002.

[153] Joshua T Trachtenberg, Brian E Chen, Graham W Knott, Guoping Feng, Joshua R Sanes, Egbert Welker, and Karel Svoboda. Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature*, 420(6917):788, 2002.

[154] Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325, 2003.

[155] Arjen Van Ooyen and Markus Butz-Ostendorf. The rewiring brain. *A Computational Approach to Structural Plasticity in the Adult Brain [Elektronisk resurs]*, 2017.

[156] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[157] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for using multiple views in information visualization. *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '00*, pages 110–119, 2000. doi: 10.1145/345513.345271. URL `http://portal.acm.org/citation.cfm?doid=345513.345271`.

[158] Chris Weaver. Building highly-coordinated visualizations in Improvise. In *IEEE Symposium on Information Visualization, 2004*, pages 159–166, Austin, TX, USA, 2004. IEEE Computer Society.

[159] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[160] PJ Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences, Harvard University*. PhD thesis, Masters Thesis, 1974.

[161] Brad Whitlock, Jean M. Favre, and Jeremy S. Meredith. Parallel in situ coupling of simulation with a fully featured visualization system. In *EGPGV*, pages 101–109, 2011.

[162] Torsten N Wiesel and David H Hubel. Extent of recovery from the effects of visual deprivation in kittens. *Journal of neurophysiology*, 28(6):1060–1072, 1965.

[163] Kong-Fatt Wong and Xiao-Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *The Journal of Neuroscience*, 26(4):1314–1328, 2006.

[164] Santiago Ramón y Cajal. *Neuron theory or reticular theory?: Objective evidence of the anatomical unity of nerve cells*. Editorial CSIC-CSIC Press, 1954.

[165] H. Yamahachi, S. A. Marik, J. N. McManus, W. Denk, and C. D. Gilbert. Rapid axonal sprouting and pruning accompany functional reorganization in primary visual cortex. *Neuron*, 64(5):719–729, Dec 2009.

[166] Esin Yavuz, James Turner, and Thomas Nowotny. Genn: a code generation framework for accelerated brain simulations. *Scientific reports*, 6:18854, 2016.

[167] Yury V. Zaytsev and Abigail Morrison. CyNEST: a maintainable Cython-based interface for the NEST simulator. *Frontiers in Neuroinformatics*, 8(23), 2014. ISSN 1662-5196. doi: 10.3389/fninf.2014.00023. URL `http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00023/abstract`.

[168] Yury V. Zaytsev, Abigail Morrison, and Moritz Deger. Reconstruction of recurrent synaptic connectivity of thousands of neurons from simulated spiking activity. *Journal of Computational Neuroscience*, 39(1):77–103, 2015. ISSN 1573-6873. doi: 10.1007/s10827-015-0565-5. URL `http://dx.doi.org/10.1007/s10827-015-0565-5`.

[169] Friedemann Zenke and Wulfram Gerstner. Hebbian plasticity requires compensatory processes on multiple timescales. *Phil. Trans. R. Soc. B*, 372(1715):20160259, 2017.

IAS Series
Band / Volume 47
ISBN 978-3-95806-577-2