RWTH Aachen University
Software Engineering Group

Spike Train Analysis in Mechanically Stimulated Rat Neural Fibers: Use Case for Design of OpenMNGlab Software

Bachelor Thesis

presented by

Tartsch, Alexander

1st Examiner: Prof. Dr. rer. nat. A. Morrison

2nd Examiner: apl. Prof. Dr. med. B. Namer

Advisor: Dr. rer. medic. E. Kutafina

Advisor: M. Sc. A. Pérez Garriga

The present work was submitted to the Chair of Software Engineering

Aachen, August 21, 2022

Kurzfassung

Eine kurze Zusammenfassung der Arbeit.

Abstract

A short abstract of this thesis.

Contents

Chapter 1

Introduction

1.1 Pain and Nociception

An important field of study in medical sciences is the study of pain. The international association for the study of pain (IASP) is the leading global organization supporting the study and practice of pain and pain relief [1]. It defines pain as "an unpleasant sensory and emotional experience associated with, or resembling that associated with, actual or potential tissue damage".

It is important to distinguish pain from the neurally coded response to potentially harmful stimuli, which is called nociception. Both nociception and pain fulfill an important warning function for the body. The definition of pain includes the internal experience, as well as the concrete stimulus response.

As a result of a lesion or disease of the somatosensory nervous system people can also experience chronic neuropathic pain. Studies suggest, that around 6-10% of people in the general population suffer from neuropathic pain [2] [3]. Treatments for this are limited and often have side effects, which makes this disease difficult to deal with [4]. We need to understand the basic principles of how the sensation "pain" is generated in order to study this further. This is especially important when pain becomes the disease itself and looses its warning function.

1.2 Neural Signalling

After a stimulus has been received by a corresponding receptor, the information needs to be passed to the brain. Signals are transmitted from sensory input to the brain via electrical signals inside of nerve fibers. Because signals inside the human body have to travel over a longer distance, the information is coded not as a continuous signal but through discrete events called action potentials or spikes [5] [6].

Spikes are seen as a binary event for information transmission, which means that the concrete spike shape does not have an effect on the encoded message. It has been shown that information is temporally encoded via patterns and firing rates of spikes [6]. Trying to figure out how exactly information gets encoded has been a long standing research question. In order to understand aspects of the neural code we want to analyze spiking patterns in C-fibers evoked by a mechanical stimulus. This might give us hints about the encoding of information within the neural network.

One method to record such data is called microneurography (MNG). It is an electrophysiological technique that can measure peripheral nerve activity in awake human subjects [7]. A needle electrode is inserted into a peripheral nerve and records its responses to stimuli. With this we only look at spikes from a single neuron, however, in reality signals are passed and processed by large quantities of neuronal clusters and what we are looking at is only a small part of encoding information [6].

We are making use of the openMNGlab framework in this thesis [8]. This is an open-source analysis framework in the Python programming language tailored for data analysis of MNG data. It was developed at the institute of medical informatics at the Uniklinik Aachen.

In this thesis we analyze single fiber recordings from the skin nerve preparation of rats from Roberto de Col and try to replicate his analysis [9] using the openMNGlab software. This data is similar to MNG data, so we can use the insights of the analysis on human patients in the future. While doing this we take a closer look at how this particular use case is handled by the software and where it can be improved. The goal of this thesis is to establish the use case of mechanically and electrically stimulated nerve data for openMNGlab and to analyze the problems of the software as they appear in the process.

1.3 Results

The import of 22 single fiber recording files were tested with openMNGlab. During this process different problems arose and workaround code was developed to solve these issues. Furthermore analysis code was developed for these recordings. This was tested on several recordings and the results for one recording can be seen in chapter 4. The code can be found in a Github repository [**?**]. With the code recordings with mechanical and electrical stimulation can now be imported and analyzed. This solves importing issues, where not all of the information necessary for analysis was included. Other users of openMNGlab were interviewed to provide ideas for future work and improvements of the framework.

Analysis methods used in [9] were replicated and concluded with the same results. Increased spiking activity of a nerve fiber, here evoked through electrical stimulation, leads to decreased response to a mechanical stimulus. In addition, other quantification approaches for the spike train data were tried and visualizations were added that can be seen in the sample analysis in chapter 4.

Chapter 2

Background

2.1 Data acquisition

A data acquisition system is a combination of software and hardware components that work together in order to control inputs and record data from different subjects. Whenever researchers want to record electrophysiological data, these systems are used. There are three systems that should be compatible with OpenMNGlab: Spike2, Dapsys and OpenEphys.

2.1.1 Spike2

Spike2 [10] is a data acquisition and analysis software produced by Cambridge electronic design limited for electrophysiological data. It is a flexible tool that can be used in different ways.

The software can record multiple channels simultaneously. An example screenshot of a recording can be seen in figure 2.1. This depicts a typical recording used for analysis in this bachelor thesis. The recording contains data from nerve fibers of rat cranial dura mater. The nerve fibers were stimulated using a electromechanostimulator applying electrical and mechanical stimulation.

First of all it contains the recorded raw signal (μV) at the bottom in channel 1. The next channel contains the temperature during the recording in degrees Celsius. In this example it fluctuates between 35 °C and 36.5 °C. In channel 3 we can observe the mechanical force that was applied to the nerve fibers. In figure 2.1 we can see the mechanical force peaking whenever a mechanical stimulation occurs to evoke a spike train.

For this experiment we want to collect the data of single nerve fibers. However, it is difficult to record just a single fiber, when recording in vitro. This is why in this experiment spike templates are applied to the raw signal to sort spikes to specific fibers. These sorted spikes are then displayed in so called wavemark channels. In this example channel 4 is such a channel, where only specific action potentials are displayed that belong to one fiber. The sorting process is done by the experimenters and is based on certain features of the action potential shape.

The electrical and mechanical stimulus events are always stored in channel 32, which is the topmost channel in figure 2.1. The two different stimuli are denoted with a different

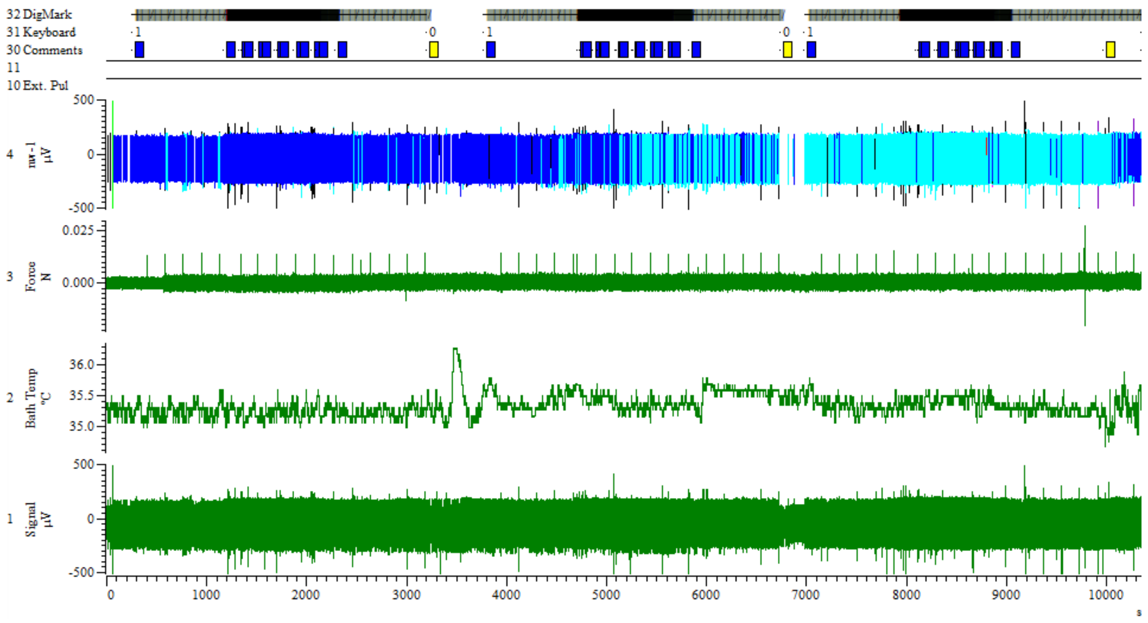


Figure 2.1: Typical mechanically and electrically stimulated recording in Spike2. There are multiple channels that display information. From bottom to top we can see the raw signal in microvolt (μV), the Bath temperature in degrees Celsius (^{o}C), the mechanical force in Newton (N) and a channel containing spikes belonging to one fiber. Another important channel is the Comments channel, which contains the experimental protocol from the experimenters. At the top the "DigMark" channel contains the timestamps for all of the electrical and mechanical events during the course of the recording.

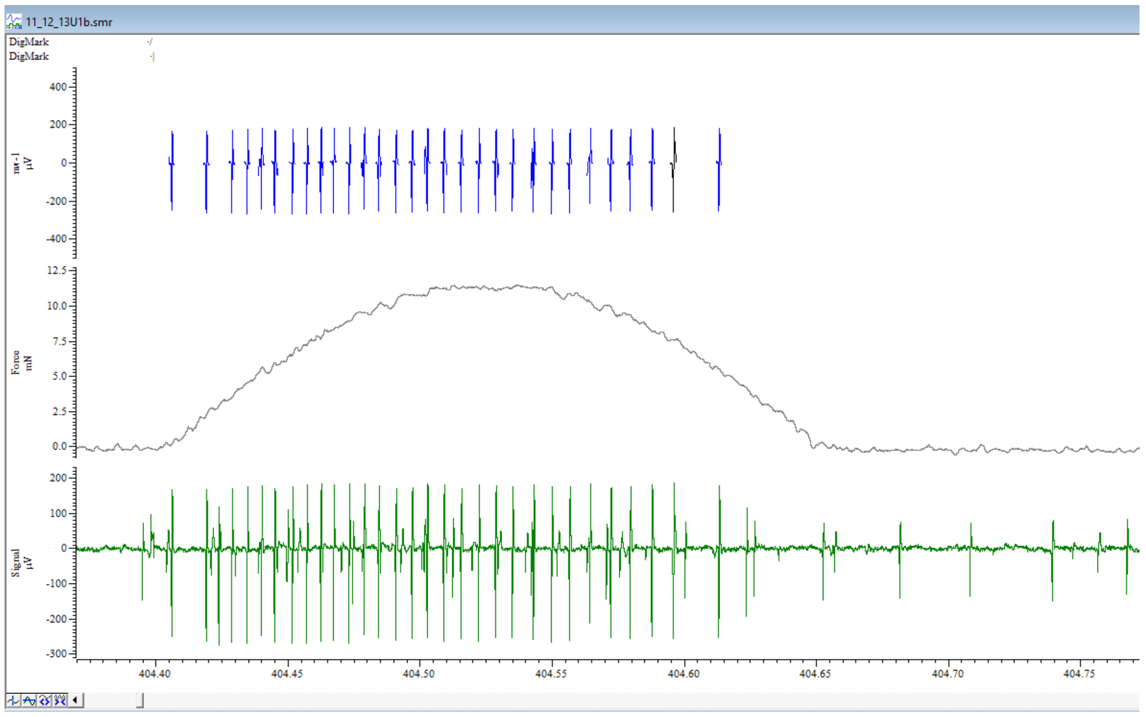


Figure 2.2: A single spike train in the Spike2 software. At the top we can see the "DigMark" channel containing a mark for an electrical and mechanical event. The channel "nw-1" contains the spikes recorded during this period in microvolt (μV). The "Force" channel displays the half sine shape of the mechanical force applied in millinewton (mN) and the "Signal" channel contains the recorded raw signal in microvolt.

symbol in this channel. Additionally there is a channel containing comments regarding the experiment. Comments can represent the experimental protocol and are given by the experimenters. In this example there are comments denoting a change in electrical stimulation frequency. In other experiments for example, these could also denote the application of certain chemicals towards the recorded subject.

A more detailed view of a single spike train is shown in figure 2.2. Here the difference in electrical and mechanical event markers in the two topmost channels are visible. Mechanical markers are represented by a slash, while electrical markers are represented by a vertical line. Another thing that can be seen here is the channel containing only the spikes. This channel is ideal for the extraction of the spikes for later analysis as there is no noise in the channel anymore and the spikes can also be interpreted as simple events with a timestamp.

2.1.2 Dapsys

The second data acquisiton package that our analysis software system needs to support is called Dapsys [11]. It is a hardware and software system that can record and analyze electrophysiological data from animal or human sources and has been mainly used for studying the peripheral nervous system. It has been in development for over 30 years since its earliest version and thus has a great history of usage in the field. The idea behind the conception was to build a system that could control stimulators and simultaneously acquire the data in real-time and display the data [12].

Dapsys offers the capabilities of path tracking and comes with the benefit of much data being available from experiments conducted with the Dapsys software. It was developed for animal experiments originally and then later adapted to also handle human MNG data. As is the case for Spike2 it also comes with a visual representation of the data. This graphical interface works real time while recording the data.

2.1.3 OpenEphys

The last data acquisition system supported by openMNGlab is called OpenEphys [13]. It is an open-source electrophysiology data acquisition system originally developed by a nonprofit based in Cambridge, Massachusetts. It sets a big focus on modularity and flexibility so that it can fit many different needs from a variety of users. As an open-source software it makes it easier to develop and share analysis and details of the experimentation process.

OpenEphys makes use of inexpensive open-source hardware such as Intan chips [13] to make it easier for small labs to get started with analyzing electrophysiological data. The heart of the software is a plugin-based graphical user interface (GUI). Here the user can add different modules to the processing pipeline such as modules for communicating with the hardware or modules for analyzing the data.

Different labs often have widely varying sets of requirements towards data analysis and therefore data acquisition systems. At the start of a project they are confronted with the question of whether to use a proprietary system or develop their own systems, which will result in much more effort before the real work can begin. Proprietary systems also often offer limited customization possibilities, which makes it harder to fit the system to the users needs. This is where OpenEphys comes in with its plugin-based software. It is designed so that everyone can mix and match the processing modules and end up with the exact analysis pipeline that they require. In addition to a wide array of already available processing modules, it is possible to develop your own modules that fit seamlessly into the structure of the system. OpenEphys is completely developed in C++ and based on a library for audio applications. This makes it easy to integrate new modules by making use of the class inheritance capabilities of C++.

OpenEphys offers many advantages such as low cost, transparency and flexibility due to the modularity, but there are also drawbacks of using the system. The start is a little harder than in proprietary systems, where there is often a straight forward way of getting started with your first experiments. The modular nature of the system might also effect the performance of the analysis, which is why commercial systems might be a better choice, if they fit the analysis needs.

OpenEphys is a system which is used by an associated research group in Bristol, who should also be able to incorporate OpenMNGlab in their analysis.

2.2 Common Spike Train Analysis Methods

Looking at the literature, there is not one definite way to analyze a spike train. Different papers use different quantifications and statistics, depending on their set of experimental data.

In the early literature attempts were made to model the probability of spikes appearing at specific times in a spike train [6]. The most common model for this is the Poisson model, which uses a Poisson process to calculate the probability of spike distribution. The book by Rieke et al. [5] also proposes a raster plot to visualize spike data featuring multiple trials with the same conditions. The response to the same stimulus did not elicit an identical response over many trials. From the time data that is used to plot this, they also plotted a post stimulus time histogram. For this they counted the average number of spikes in bins of a specific time (10ms). This gives the firing rate as a function of time.

The paper from Uebner et al. [9] that produced the data that this thesis is about used a different quantifier. In this paper the effect of background stimulation on a mechanical stimulus that did not change was studied. To quantify this effect, they used the peak firing frequency that averages the 5 highest instantaneous frequencies of spikes per spike train. This was the main indicator for changes in the spike trains and therefore it is also incorporated in the analysis in this thesis.

2.3 Spike Train Analysis Tools

2.3.1 FieldTrip

Fieldtrip [14] is a MATLAB toolbox developed at the Radboud University, Nijmegen, the Netherlands and offers a wide variety of analysis functions. It can analyze MEG, EEG, and iEEG and is an open-source software that has been in development since 2003. Its main strengths lie in the analysis of invasive and non-invasive electrophysiological data.

It provides over 100 high-level and 800 low-level functions that can be used both by experimental users as well as developers. It does not feature a graphical user interface, but instead focuses on providing direct access to the high and low-level analysis functions that can be used in the command line or in scripts. This increases flexibility and is done because FieldTrip is not meant to be an application but a set of tools that can be mixed and matched for different requirements.

With this approach the user needs to be familiar with MATLAB before starting their analysis, but after a potential initial time invest the flexibility of this approach yields great advantages.

The analysis functions are meant to be combined and used as a sort of analysis protocol, where the output of functions is used to compute the next results. An example pipeline is lined out in the paper [14]. The first step after loading a data set is to choose a data segment to be analyzed by setting the boundaries of the segment. In order to free the data from noise and artifacts that could compromise the analysis. FieldTrip provides the corresponding functions for automatic removal of artifacts. As a next step the user can run a preprocessing function that loads the specified data into the MATLAB workspace and readies it for further analysis. Then the user can choose which specific analysis they want to perform. There is also the possibility of source reconstruction to get a visual representation of the source of the data.

An important feature of FieldTrip is the ability to save and reuse intermediate results. After each step in the analysis there is the possibility to save the current results for later use before further altering the data. This can be useful in many situations where the user wants to retrace certain steps in the analysis.

For visualization of results users can make use of standard MATLAB functionalities to plot the numeric results of FieldTrip analyses.

As an open-source software FieldTrip is also meant to be contributed to by developers who see opportunities for improvement. As opposed to other GUI-based software FieldTrips focus on having the direct access to functions lends itself to development and contributions from various different experts, which only enhance the product.

This software package is very useful and widely used in the field, however it does not lend itself to our specific needs. The main problem with this software is its programming language. It is a MATLAB toolbox, however it would be preferred to use a software package in Python or another programming language that slots better in the already existing structure that is used at the chair for medical informatics.

2.3.2 Elephant

Elephant [15] is a Python module which offers high-level analysis functions for electrophysiological data. It also features functions that are designed specifically for spike trains. It does provide functions for high-level analysis, but is lacking when it comes to very basic functions and quantifiers for spike trains.

Its focus on highly specified analysis tools makes it not viable in our use case. In this thesis we want to start with the basic signal from the spikes. Then the goal is to try and quantify spike trains, starting with basic measures as number of spikes for example. This is not something that Elephant provides. Once this ground level analysis is done we can think about the more advanced analysis provided by such tools as Elephant.

2.4 openMNGlab

The previously presented analysis tools have their uses in different use cases, but are not suitable for this thesis for a variety of reasons. There are multiple data acquisition systems that are used at the institute of medical informatics (IMI) that each produce different file formats. The three main systems that need to be handled are the previously described Spike2, Dapsys and OpenEphys.

We need a tool that that has the capability to load files from these different data acquisition systems, put them in a compatible format and analyze them further. FieldTrip does not offer these kinds of capabilities as well as it being a MATLAB package, which would not be ideal for fitting into the rest of the analysis systems at IMI, which are based on Python. Elephant is a Python tool, but is lacking the importing tools that would be required for the software solution.

For this reason IMI has started to develop their own software framework in Python called openMNGlab [8]. This framework aims to provide a solution for dealing with different experimental file types and combine them into a single usable format. In addition it aims to provides analysis capabilities for microneurography data.

OpenMNGlab was started as a project of IMI and was initially developed by Fabian Schlebusch who developed openMNGlab 1.0 and set up the basic structure and developed the importers necessary for the file formats that we require. However, the 1.0 version of the framework, being an early version, still came with a few issues. The biggest one being the functionality of the importers. The software was not capable to import the raw experimental files from the Spike2 and Dapsys systems. An additional extraction step was required to create csv files that could then be imported into openMNGlab.

In the beginning the goal of having everything in one format was not a priority compared to getting a working analysis for the different files. This resulted in analysis functions that were not usable for every recording that should have been analyzed.

2.4.1 Neo

To help with these issues, Neo [16], a Python package for representing electrophysiological data, was integrated into openMNGlab. This package provides a way to model electrophysiological data in a hierarchical structure that becomes the new basis of openMNGlab. From now on the data will be modelled with Neos hierarchical structure.

Neo also comes with built in importer functionalities for many different files formats and templates to develop importers for new file formats. These importing tools help to solve the previous importing issues in openMNGlab. This enables us to import raw Spike2 files directly, which makes the analysis process much less cumbersome as one step in the analysis pipeline gets eliminated. When using the importers provided either through Neo directly or via its templates, the imported data immediately has a uniform structure, regardless of its origin.

Neo does not provide analysis on its own, but we can now base all of our analysis on the structure it provides (that will from now on be referred to as Neo structure).

Cooperation with other software

The importer for Spike2 files was already developed for openMNGlab using the Neo package and therefore many experiments that were recorded with the Spike2 software can be analyzed with the framework. The same can be said for OpenEphys files, since Aiden Nickerson, a collaborator in Bristol, has implemented a corresponding importer. OpenMNGlab already features an importer for Dapsys recordings, which makes many experiments conducted with this system readily available for analysis. The difference to Spike2 however, is that the importer can not deal with the raw recording files, but requires an extra step from the user. A user needs to export the raw data he wants to analyze as csv files from the Dapsys software, before he can import them into openMNGlab. This process is especially cumbersome when dealing with a larger number of recordings. In the future the importer functionality of openMNGlab should be improved, so that it works on the raw Dapsys files in order to make the analysis workflow easier. This is especially useful since there are many experiments recorded with Dapsys that would be available for analysis.

The Matlab functions offered by FieldTrip, while similar from a functionality point of view are not compatible with our Python environment. We can, however, take a look at the functions of FieldTrip and take inspiration for similar functionality in openMNGlab. The case is different for Elephant. While the functions may be too advanced for the analysis done in this bachelor thesis, they might become useful in future work. With Elephant also being a Python module, the compatibility to openMNGlab is much higher and the possibility of combining functions from each package in the future might be worth keeping an eye on.

2.5 Experimental Data

This thesis is based on the analysis of Uebner et al. [9]. In this paper they analyzed the activity dependent axonal conduction velocity slowing in slowly conducting meningeal afferents. They studied the effect of preceding action potential activity on mechanical stimulus response.

The results show that there is a significant decrease in peak firing frequency after the fiber was previously electrically stimulated with a frequency of over 2 Hz. The analysis code that was written for this thesis replicates part of the analysis from the paper and comes to the same conclusions about the decrease in peak firing frequency and overall spiking activity.

Chapter 3

Methods

3.1 Software

In this thesis recordings of mechanically and electrically stimulated rats were analyzed using the same quantifiers as in the paper by Uebner et al. [9]. This was supposed to be done using the openMNGlab framework [8]. The goal was not to provide a fully functioning analysis system as part of this thesis, but rather comment on the process of using openMNGlab and provide ideas for any issues occurring during the analysis. To get a better understanding of openMNGlab, other users of the framework were interviewed to get their perspective. This chapter is split in two parts that deal with the software development and spike train analysis respectively.

The first step in the process to understanding the problems with the software was to use the software. Doing so, testing for potential issues was done while also doing work towards the analysis. While using the software various different errors and issues of varying size and importance came up. After using the software and getting to know the framework for the use case of electrically and mechanically stimulated data, other users were interviewed about the framework to get an idea of other potential issues with the software that did not appear during the previous work for this thesis. This resulted in a comprehensive list of different users needs and issues regarding the software framework. From this point of view solutions and ideas for improvements and development of openMNGlab can be proposed.

3.2 Spike Train Analysis

3.2.1 Data

An integral part of the analysis is of course the data itself. Because human nerve data is hard to obtain, we can also use animal data instead. Animal data is usable as a proxy, because we can observe the nerve fibers in vitro but can better separate one single nerve fiber from others. In human data an additional step of fiber separation is necessary to differentiate between individual fibers. We can use the same experimental protocols on animals as we would on humans. This way we can understand firing patterns of spikes and quantify them. The results can then be applied to human data.

In the case of this thesis, we are using the data from wistar rats. The data was recorded from 2011 to 2012 by Roberto de Col and was published in a paper by Uebner et al. [9]. The goal of the paper was to evaluate the effects of previous spiking activity on the nerve fiber response to mechanical stimulation.

The experiments were done in vitro on peripheral nerve fibers. The fibers were mechanically and electrically stimulated via a custom made electromechanostimulator. The nerve activity was recorded using an electrode. The electrical stimulation consists of small electrical pulses that come in a controlled frequency. The mechanical force is applied in a half sinusoidal shape. In the recordings the mechanical force that is applied throughout stays at approximately the same level, but there are exceptions where the mechanical stimulation changes in amplitude and length during one recording.

3.2.2 Spike Train Definition

Earlier in this thesis we covered the basics of electrical signals in nerve fibers and action potentials. In this thesis we are, however, not interested in single spikes but in clusters of them. For this it is important to define what we mean when we speak of spike trains.

In theory any number of spikes could be called a spike train. What we are interested in is the short amount of time after a stimulus event where there is increased spiking activity, see figure 2.2. In the data provided by Roberto de Col there is mechanical and electrical stimulation [9]. The mechanical stimulation, which is approximately 250 milliseconds long, leads to noticeably increased spiking activity. It is this spiking activity that we want to analyze and quantify and to have a unified way of speaking about this activity we call these clusters of spikes spike trains. We consider the mechanical stimulus the start of the spike train. This is marked in the event channel of the data.

A spike train is considered finished after the increased spiking activity has normalized. For the analysis in this thesis the maximum duration of a spike train is considered to be 500 milliseconds. Each spike occurring within 500 ms of the mechanical stimulus gets counted towards the spike train. The duration of the spike trains in the recordings ranges from 150 milliseconds to 500 milliseconds.

3.2.3 Spike Train Analysis: Quantifiers

Now that we had a look at the data that we have available, it is time to discuss the quantifiers that are used for the analysis of the data. Starting out with raw data the first and perhaps most obvious quantifier that can be computed is the number of spikes in a spike train. It is a basic measure and can give information at a quick glance about the spike train. Of course this is not as expressive as other quantifiers and does not necessarily provide us with much context, but it can give hints at less or more active nerve fibers for example.

Another quantifier mentioned in the literature is the mean firing rate. This is calculated by dividing the duration of the spike train by the number of spikes and gives an idea of the frequency in which the nerve fiber fires at this moment.

The next quantifier included in the analysis is the peak firing frequency. This is included, since it is one of the quantifiers mentioned in the paper by Uebner et al. [9]. The peak firing frequency is calculated for each spike train and details the concentration of spiking activity. During development two different implementations of calculating the peak firing frequency were tested. With the first version, the instantaneous frequencies of a spike train were sorted by value and the 5 highest frequencies were averaged. This, however, did not give a representative look on a single section of spikes. The second version switched to a sliding window implementation, where the average instantaneous frequency over a sliding window of width 5 were calculated and the highest average was chosen as the peak firing frequency.

Chapter 4

Results

4.1 Software

4.1.1 Data Structure

There are different data structures used in the analysis. This comes as a result from the development process.

Neo Structure

Neo models electrophysiological data in a hierarchical structure, which is depicted in figure 4.1. On the lowermost level we start with different kinds of data objects. An *AnalogSignal* is regularly sampled data and can contain multiple channels. An *IrregularlySampledSignal* is similar, but does not feature a regular sampling interval. A *SpikeTrain* object contains time point data with the information when action potentials occur. These *SpikeTrain* objects can cover a large interval in time such as the whole duration of the recording and thus is a little different from the definition of a spike train that is otherwise used in this thesis, which I have described in the previous chapter. *Events* in Neo point to distinct points in time and can be used to mark stimulation events for experiments with animals for example. *Epochs* function similar to events, but cover a duration instead of just points in time.

On the next layer up there are containers for grouping all of these lower level objects. The first of these containers is a *Segment*. This groups data that was simultaneously recorded. On the topmost layer there are *Blocks*. One *Block* can contain multiple *Segments* which in turn can contain multiple data objects.

In case of this Bachelor thesis we are dealing with recording files from animal experiments. When importing one such file, the resulting Neo structure looks as follows: On the top level each file contains a *block*. In our case these *blocks* contain only one *segment*, since the recordings feature a single continuous signal without interruptions.

Custom Structure

The analysis notebook is still using code from version 1 from openMNGlab where the Neo structure was not yet integrated. The importer from version 1 delivered the data in a

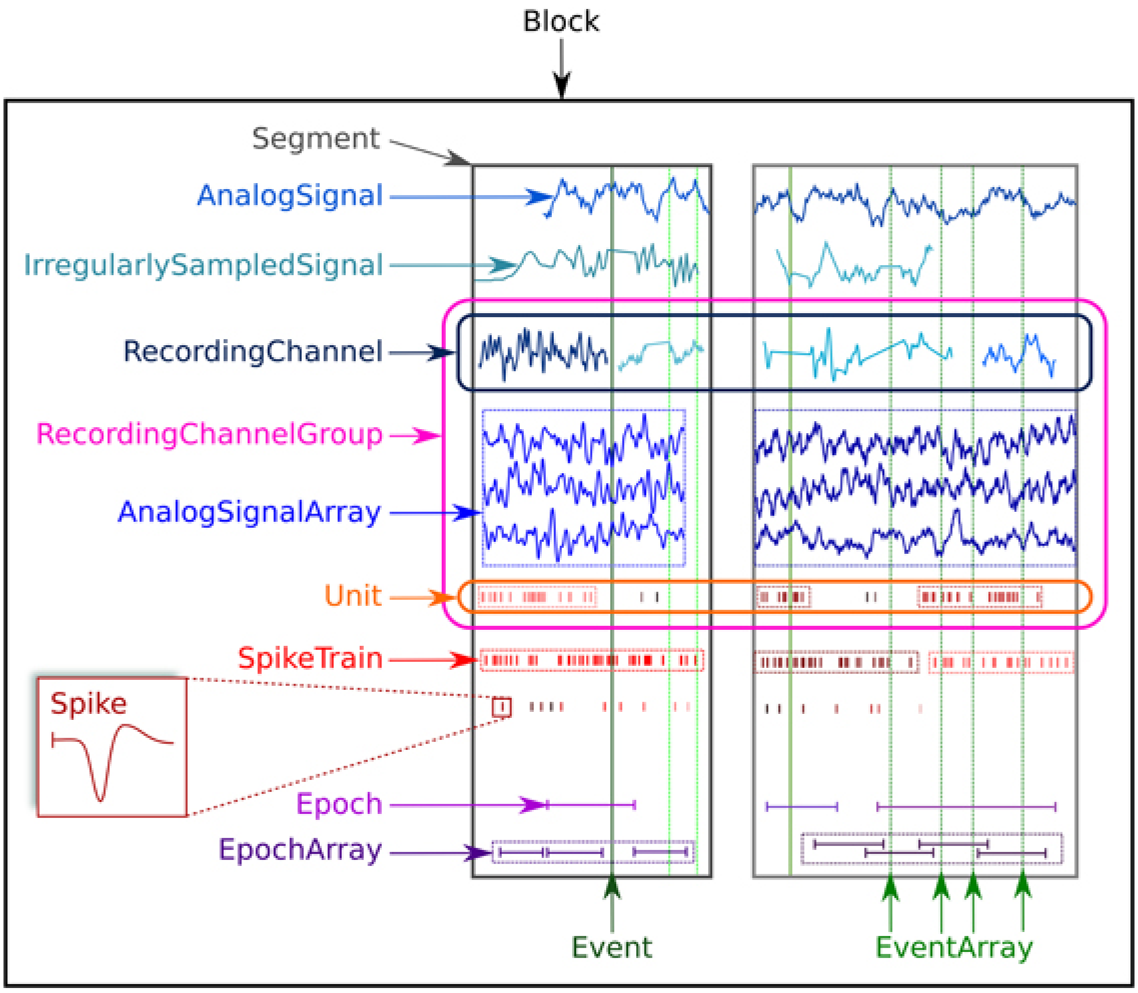


Figure 4.1: An illustration of the data types supported by Neo and their grouping into containers taken from [16]. AnalogSignals and IrregularlySampledSignals contain regularly and irregularly sampled data respectively. SpikeTrain objects contain a set of spikes from the same source. Events consist of a label and a time of occurrence and Epochs represent a period of time in the experiment. The container object Segment can contain multiple data objects that share the same clock. A Block is the top-level container and can contain multiple Segments.

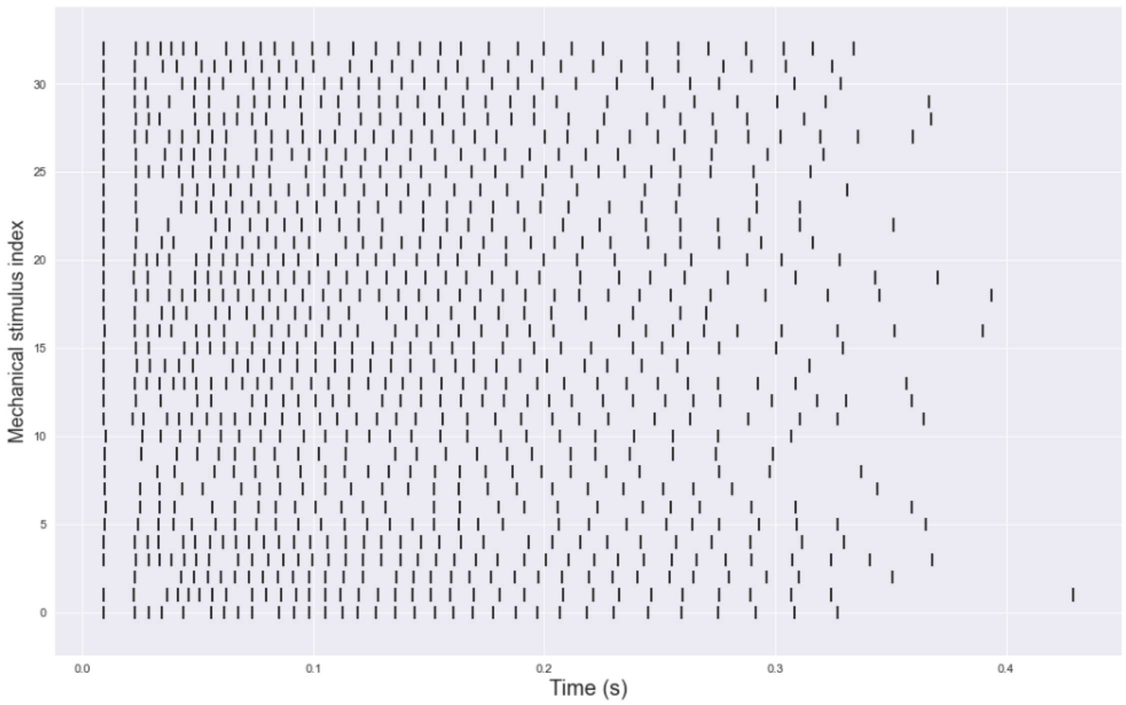


Figure 4.2: Event plot detailing spiking activity for one recording. The x axis notes the time since the start of the last mechanical stimulus. Each row on the y axis shows the spikes of one spike train. Each line represents one spike in the recording.

custom data structure.

At its core the custom data structure models the data hierarchically, but the objects look a little different to the ones from Neo. At the base level there are data objects *ActionPotential*, *MechanicalStimulus*, *ElectricalStimulus* containing information corresponding to their respective names. These objects are all part of an overlying object type called *signal_artifact*.

On the top level we get an object called *recording* when importing data with this method. One such *recording* then contains the lists *el_stimuli*, *mech_stimuli*, *actpots* which are lists containing objects of types *ElectricalStimulus*, *MechanicalStimulus* and *ActionPotential* respectively. Additionally there is a *raw_signal* object which contains the raw signal in the form of arrays.

4.1.2 Development Process

To analyze any data a jupyter notebook is used which contains all the relevant code. The base of the analysis notebook started with the work of Radomir Popovich, who also worked with spike train data for IMI. He had developed a jupyter notebook based on a custom import of spike train data extracted from the Spike2 software as a csv file. From this he extracted the spike trains and created figures such as event plots as seen in figure 4.2. This is a raster plot that depicts the spike trains of a single recording. Each row represents one spike train. The x axis represents the time since the last mechanical stimulus starting at 0. These figures give a good overview with one glance over the spiking activity during mechanical stimulation for recordings. Also included in this notebook was a way to filter the action potential channel so that only the relevant spikes for our spike trains remain. This was done by checking an interval of a specific duration after a mechanical stimulation event occurred for spikes and saving those in designated lists, which is still the method we use to extract the spike trains.

Starting from this point there were a couple of iterations trying to import the data, which can be seen in figure 4.3. The first iteration used the previously described custom import. The Spike2 data was exported into csv files that were then imported in the code. In addition version 1 of openMNGlab was used to import the spikes and mechanical events. This importing technique is rather slow, however, and is missing the electrical stimulation information, which is vital for the analysis.

When Neo was integrated into openMNGlab, it brought with it new ways of importing data. With this addition, the Spike2 files could be imported directly into the code, which resulted in fewer intermediate steps and is also much faster. The issue with the Neo importer of Spike2 data is that the spike sorting gets lost. The experimental researchers can apply spike sorting in the Spike2 software and create wavemark channels which feature only certain spikes. These channels do not get imported with the Neo importer and so the information which spikes are relevant and which spikes can be ignored gets lost. Another thing that is missing from the Neo import is the mechanical force channel. This means we are missing information about the duration and amplitude of the mechanical stimulus with this importing method.

In the end we combined all of the importing methods used in the previous two iterations. With the Neo import we gather the electrical events and stimulation frequency, with the custom import code from the very beginning the mechanical force channel is imported and with the import from version 1 of openMNGlab we receive the mechanical event timestamps as well as the spikes.

This is not an ideal solution, as three different importing methods are needed that result in three different data structures. The import of the csv file also slows down the analysis considerably, but for the purposes of this thesis and the analysis of only a few files this solution sufficed.

Issues with Importing and their Solutions

There is a channel in the Spike2 files where experimenters can leave comments while experimenting. This is used to make notes on the experimental protocol of the recording in many cases. In the files that were provided for this thesis there were a couple of files where the character μ was contained in this channel. While importing the corresponding files an error appeared, because the importer did not recognize the Greek letters. This problem was circumvented by deleting the comment channels in those files, as they did not contain relevant information for my analysis. This is generally not a viable solution, since the comment channel is vital for many experiments and includes important information regarding the experimental conditions. Therefore, in the future openMNGlab needs to implement a fix to the importer, so that the comment channel can be included properly.

4.1.3 Use Cases

To end the first part of this chapter different users were interviewed to share their use cases of openMNGlab. Specifics of each use case are detailed and remarks are given on the

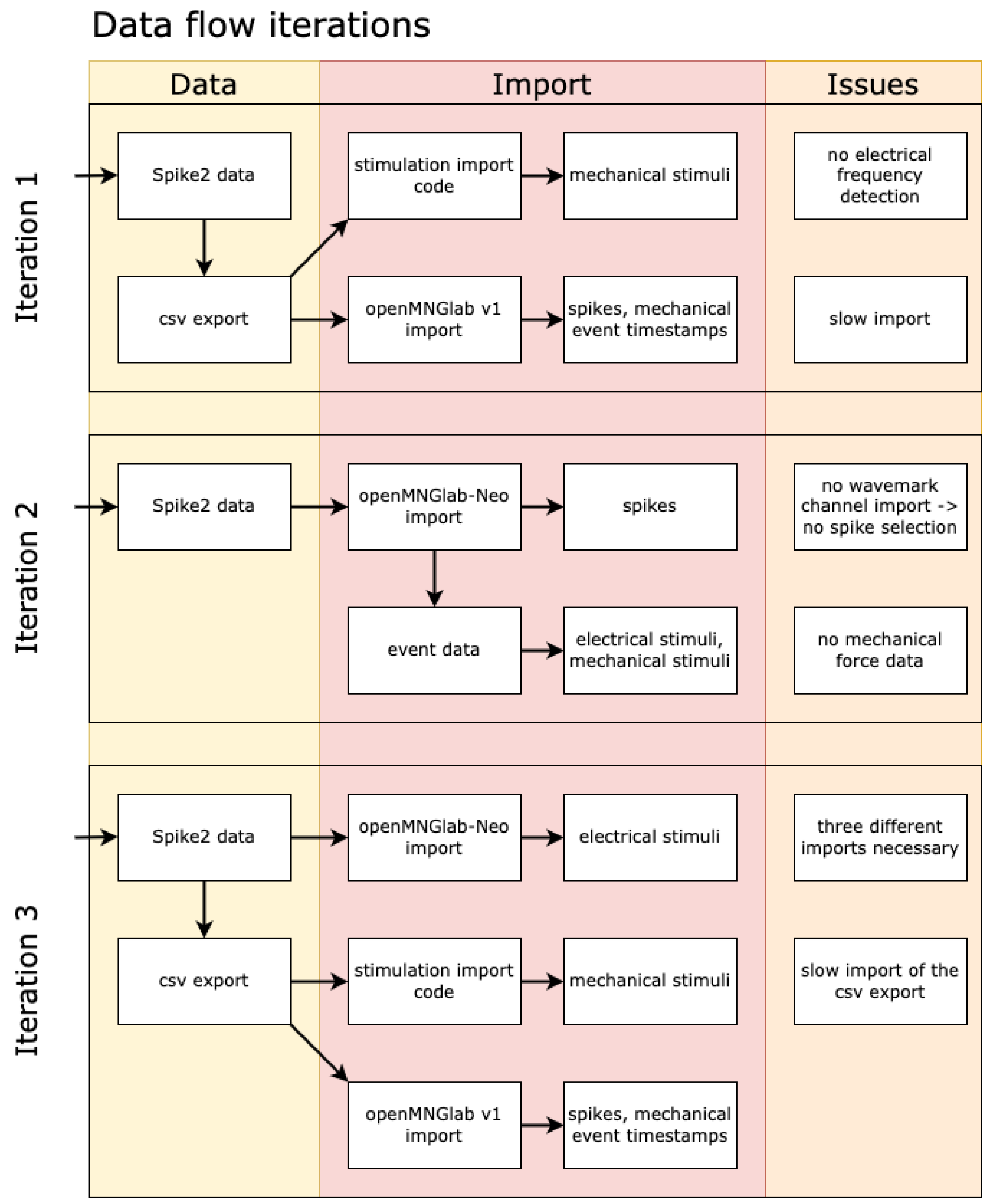


Figure 4.3: This diagram shows the different iterations that the data flow went through. The first iteration uses a custom csv import and the first version of openMNGlab to import mechanical stimuli and spikes. The second iteration uses the Neo importer from openMNGlab to import spikes and stimulus events. The third iteration combines the first two approaches to import spikes, electrical and mechanical stimuli.

current state of openMNGlab and the potential improvements necessary for a smoother experience for all users.

One important feature of openMNGlab that is needed in every use case is the importing of new data. For new users it is important to be able to load sample data sets and get a feel for the framework by getting to know a few test analysis functions. For students or experimental researchers it is key to be able to load multiple files for a quick overview or even full analysis.

Because we are dealing with different sources when it comes to experimental data, which all need to be handled in a different way, this is where many issues can occur. When going through the use cases we will see where we encounter difficulties regarding the data import.

Student 1

The first user is a student who does data analysis on mechanically and electrically stimulated Spike2 data. The goal is to perform latency analysis for spikes. The raw Spike2 files feature a lot of information, not all of which is always needed. In this case, all that is required to analyze the latencies of the spikes are the timestamps of the spikes itself and the timestamps of the stimulation events.

For the Spike2 software this means that we need to extract the "DigMark" channel which contains the event information as well as the corresponding wavemark channel which contains the information on the already presorted spikes. The relevant information can be imported using the importing function of openMNGlab. The data is then used to calculate the latency for the spikes corresponding to the latest electrical stimulus event as well as calculating the spike count.

Potential problems: The student started the analysis before the first version of the framework was finished and therefore used her own custom analysis notebook. This means that the results were not in a standard format that can be reused with the current version of openMNGlab. With the inclusion of Neo the openMNGlab framework now has the capabilities to import the data and deliver it in a standard format.

This does not work, however, once the experimental data features both electrical and mechanical stimulation. In recordings with both electrical and mechanical stimuli, the event markers for those stimuli share the same channel in Spike2. Although they can be distinguished in Spike2 itself, the imported channel in the Neo format does not distinguish between different event types. This is a problem which has to be addressed in future work.

Student 2

The second user is a student who uses mechanically and electrically stimulated Spike2 data. Their goal is to analyze spike trains resulting from mechanical stimulation. For this not all of the information contained in the raw Spike2 file is needed. The event information for mechanical and electrical stimuli as well as the mechanical force information for details of the mechanical stimuli is required. The event information can be found in the "DigMark" channel of the Spike2 file and gets extracted by the Spike2 importer of openMNGlab. The two types of events are contained in the same channel and can only be separated with contextual information from the experiments. A mechanical stimulus in these recordings always appears together with an electrical stimulus. Therefore, a mechanical stimulus can be detected by the temporal distance to other stimulus events. To get more information about the mechanical stimulus, such as smplitude and duration, the user also needs to import the mechanical force channel. It contains a continuous signal of the applied force. In addition user two also needs the information from the wavemark channel to get the spike timings.

Potential problems: The "DigMark" channel with two different event types can lead to potential problems. In this particular case they can be distinguished using contextual information, but this might not be the case for other types of experimental data. In addition these experiments introduce problems when it comes to importing the mechanical force channel. The current version of openMNGlab does not import this channel. If this information is needed for analysis, the user has to import it in another way, which leads to non-standard data structures. The inclusion of the mechanical force channel in the importer of openMNGlab is requires future work.

Student 3

The third interviewed user of the framework works on a student project to analyze Spike2 data in which chemicals were applied to the test subject. In contrast to the other two users, this user started with the project when the current version of openMNGlab with the integration of the Neo module already in place. This means she made use of the new importing functions from the beginning. In addition to the spiking activity, as in the previous use cases, this user also needs information regarding the chemicals that are applied. She requires the timings and doses of the specific chemicals. From this she specifies a time frame where she wants to monitor the spiking activity that results from the application of the chemicals.

Potential problems: The analysis of chemically stimulated data does bring with it its own new set of problems. There is no dedicated channel for chemical information in the Spike2 software. For this reason the chemical protocols are given in the "Comments" channel in Spike2. The experimenter creates a comment every time a new chemical is applied and fills it with contextual information. The current version of openMNGlab does not import the "Comments" channel, so this is something that has to be addressed in future work. The Neo structure has the data type "event" which features a timestamp and a label that could represent the comments.

Experimental Researcher

Another big group of potential users for openMNGlab are experimental researchers who produce and work with electrophysiological and microneurographical data often. They will be a big part of the user base of the framework because of the nature of their work. For them it is important that the new data sets can be easily and quickly loaded and overview statistics can be displayed. This group of people will most likely be working with a data acquisition system primarily. As we have seen in some examples already it may come to translation errors between data acquisition and openMNGlab. It is therefore important for the experimental researchers to know this framework and its quirks for a smooth collaboration of these different systems.

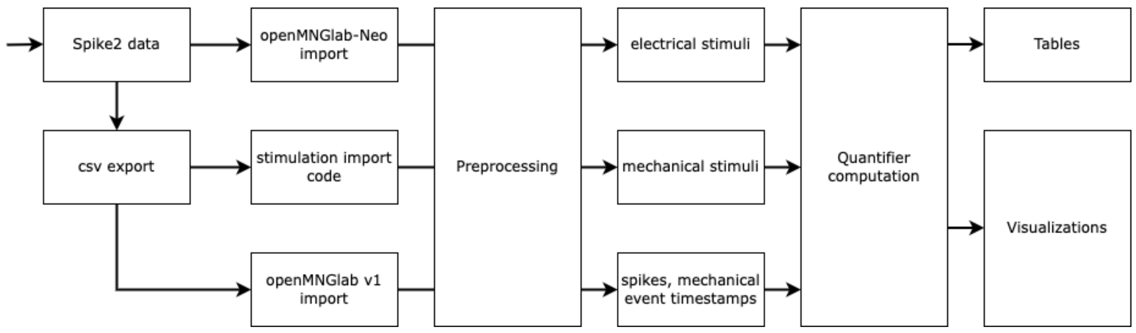


Figure 4.4: Schematic version of the analysis pipeline. The raw data gets imported with different techniques, as we saw in figure 4.3. After handling the different input structures, we receive the spikes and stimuli. The quantifiers described in chapter 3 are computed and can be visualized in tables and diagrams described in detail in this chapter.

4.1.4 Finished Analysis Pipeline

The software development process resulted in a jupyter notebook, which contains the analysis pipeline used for this thesis. A schematic version of this pipeline can be seen in figure 4.4. For a more detailed view on the different functions, the code can be found in a Github repository [**?**].

Importing Data

The first part of the notebook is the import of the data, which consists of three separate importing steps. First the Neo importer from the current version of openMNGlab is used to extract the information about the underlying electrical stimulation. In a second step we use the importer from the version 1 of openMNGlab to get the spike timings as well as the information about the mechanical stimuli. Lastly, to get the information about the mechanical stimuli, such as amplitude and duration, from the previously described custom import.

After the importing steps we end up with separate data structures, each containing part of the experimental data. From the first importing step, we get the event information about the electrical stimulation in the Neo format. The mechanical stimuli with its physical properties such as amplitude and duration, as well as the spike timings, are contained in a custom data structure. For the future it would be ideal to develop openMNGlab so that everything can get imported with the Neo importer in one single step and everything is contained in a unified data structure.

Preprocressing

The next part of the notebook contains internal processing steps to prepare the data for easy representation and quantification. Making use of the early versions of openMNGlab and analysis attempts, the event plot for the recording file gets computed and visualized.

Computing Quantifiers

After preparing the data, quantifier computation can take place. This is where the computations of the quantifiers described in chapter 3 happens. After all the computations are done, we end up with lists of quantifiers for each spike train in the recording. These are the main result of the analysis pipeline. They can be further processed and displayed in diagrams or be saved for later use. The data structure in which the quantifiers are saved in the analysis is a custom structure. Once openMNGlab can import and handle this type of experimental data fully, it can also be saved in a standard way such as hdf5.

Visualizations

An important aspect to understanding the results of the analysis are the visualizations. Raw lists of data are not as easily readable and do not offer great insights at a first glance. Therefore, we compute different diagrams from the quantifiers to help with this issue. These diagrams range from event plots that we have seen before to figures comparing different quantifiers for a recording. They will be explained in detail in the following sections.

4.2 Spike Analysis

We have analyzed 22 recording files featuring mechanical and electrical stimulation. An overview for the data can be found in table 4.1. Here we see the number of spike trains in each file and the average spikes per train as well as the average spike train duration. More information about the specific recordings is following in the next section.

4.2.1 Experimental Protocol

As mentioned in previous chapters, the files analyzed in this thesis feature electrical and mechanical stimulation. The fibers are mechanically stimulated with a custom-built electromechanostimulator [9]. For each fiber the mechanical threshold was determined as the force which would evoke a spike in 50% of the time. The mechanical force was set as double the threshold for the duration of the experiment.

During their research Uebner et al. found that a half sinosoidal stimulus profile works best for the experiments. The attributes for this type of stimulation, like amplitude or duration, are constant over the course of single recordings, but may vary between recordings. In addition to the mechanical stimulation there is also electrical stimulation, which is applied as electrical impulses with a certain frequency.

In the experiments there is a first period of 15 minutes with an electrical stimulation frequency of 0.1 Hz. Then follows a second period of increased stimulation frequency. The increased frequency is either 2, 4 or 5 Hz. The third period features 0.1 Hz stimulation frequency again. During each of the 3 periods 5 mechanical stimuli are applied with a recovery time of 3 minutes in between stimuli. After the second period there is a period of 0.5 Hz stimulation that lasts for 3 minutes.

Each recording features the protocol at least once. It was repeated in many of the files with a resting period without any stimulation in between protocols. The files in Table 4.1

File number	Number of trains	Avg. spikes per train	Avg. train duration (s)
1	17	12.53	0.38
2	34	16.62	0.39
3	37	20.03	0.40
4	12	4.67	0.28
5	11	9.18	0.25
6	22	8.55	0.12
7	17	10.82	0.38
8	16	7.38	0.40
9	28	12.93	0.37
10	35	8.83	0.35
11	37	15	0.41
12	31	13.45	0.38
13	18	10.28	0.24
14	22	35.64	0.44
15	32	13.91	0.13
16	32	25.53	0.39
17	33	30.45	0.23
18	33	29.30	0.31
19	31	11.74	0.38
20	48	10.85	0.40
21	51	22.8	0.24
22	50	12.16	0.36

Table 4.1: This table shows basic numbers for each of the 22 analyzed recording files: the number of spike trains, the average number of spikes per spike train and the average spike train duration in seconds for each recording.

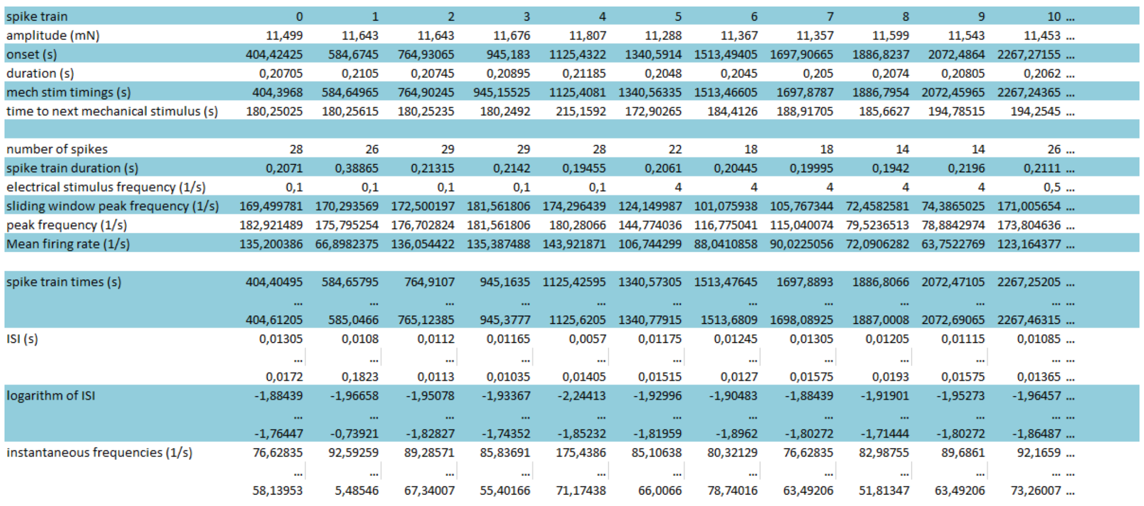
are ordered according to the frequencies of electrical stimulation that occurred in the recordings. Files 1-3, 4-12, 13-14 contain only frequencies of 2, 4, 5 Hertz respectively. Files 15-20 contain both 2.0 and 4.0 Hertz frequencies and files 21-22 contain all three types of increased frequencies.

4.2.2 Sample Analysis

To show the results of the analysis in more detail we will look at recording 21 as an example and demonstrate the outcomes. This will provide details of the quantifiers as well as present possible visualizations.

Table of Values

When applying the analysis notebook to a recording, the first output we get is a big table containing the quantifiers as well as the raw data of the mechanical stimulus amplitude, duration and spike timings. A sample table can be seen in table 4.5, which shows a screenshot of the output table for recording 21. The table contains the mechanical stimulus information for each train. This includes the amplitude, duration and timestamp of the corresponding stimulus. In the figure this information is contained in the upper third of the table.



spike train	0	1	2	3	4	5	6	7	8	9	10	...
amplitude (mN)	11,499	11,643	11,643	11,676	11,807	11,288	11,367	11,357	11,599	11,543	11,453	...
onset (s)	404,42425	584,6745	764,93065	945,183	1125,4322	1340,5914	1513,49405	1697,90665	1886,8237	2072,4864	2267,27155	...
duration (s)	0,20705	0,2105	0,20745	0,20895	0,21185	0,2048	0,2045	0,205	0,2074	0,20805	0,2062	...
mech stim timings (s)	404,3968	584,64965	764,90245	945,15525	1125,4081	1340,56335	1513,46605	1697,8787	1886,7954	2072,45965	2267,24365	...
time to next mechanical stimulus (s)	180,25025	180,25615	180,25235	180,2492	215,1592	172,90265	184,4126	188,91705	185,6627	194,78515	194,2545	...
number of spikes	28	26	29	29	28	22	18	18	14	14	26	...
spike train duration (s)	0,2071	0,38865	0,21315	0,2142	0,19455	0,2061	0,20445	0,19995	0,1942	0,2196	0,2111	...
electrical stimulus frequency (1/s)	0,1	0,1	0,1	0,1	0,1	4	4	4	4	4	0,5	...
sliding window peak frequency (1/s)	169,499781	170,293569	172,500197	181,561806	174,296439	124,149987	101,075938	105,767344	72,4582581	74,3865025	171,005654	...
peak frequency (1/s)	182,921489	175,795254	176,702824	181,561806	180,28066	144,774036	116,775041	115,040074	79,5236513	78,8842974	173,804636	...
Mean firing rate (1/s)	135,200386	66,8982375	136,054422	135,387488	143,921871	106,744299	88,0410858	90,0225056	72,0906282	63,7522769	123,164377	...
spike train times (s)	404,40495	584,65795	764,9107	945,1635	1125,42595	1340,57305	1513,47645	1697,8893	1886,8066	2072,47105	2267,25205	...
	...	...	...	...	...	...	...	...	...	...	...	...
	404,61205	585,0466	765,12385	945,3777	1125,6205	1340,77915	1513,6809	1698,08925	1887,0008	2072,69065	2267,46315	...
ISI (s)	0,01305	0,0108	0,0112	0,01165	0,0057	0,01175	0,01245	0,01305	0,01205	0,01115	0,01085	...
	...	...	...	...	...	...	...	...	...	...	...	...
	0,0172	0,1823	0,0113	0,01035	0,01405	0,01515	0,0127	0,01575	0,0193	0,01575	0,01365	...
logarithm of ISI	-1,88439	-1,96658	-1,95078	-1,93367	-2,24413	-1,92996	-1,90483	-1,88439	-1,91901	-1,95273	-1,96457	...
	...	...	...	...	...	...	...	...	...	...	...	...
	-1,76447	-0,73921	-1,82827	-1,74352	-1,85232	-1,81959	-1,8962	-1,80272	-1,71444	-1,80272	-1,86487	...
instantaneous frequencies (1/s)	76,62835	92,59259	89,28571	85,83691	175,4386	85,10638	80,32129	76,62835	82,98755	89,6861	92,1659	...
	...	...	...	...	...	...	...	...	...	...	...	...
	58,13953	5,48546	67,34007	55,40166	71,17438	66,0066	78,74016	63,49206	51,81347	63,49206	73,26007	...

Figure 4.5: Screenshot of the table of values in excel. The table contains quantifiers and raw data of the spike trains in a recording. At the top there is information about the mechanical stimulus (e.g. amplitude, duration and onset). In the middle there are single value quantifiers such as peak firing frequency or mean firing rate. At the bottom there is raw data: spike times, inter spike intervals and instantaneous frequencies.

In addition the table features the single value quantifiers for the spike trains that were described in the previous chapter. These include the peak firing frequency, spike count and mean firing frequency and can be found in the middle third of the figure. Lastly the table also features raw data such as the spike timings as well as inter spike intervals and instantaneous frequencies. These are depicted in the lower third of the figure in an abbreviated version for the sake of visibility.

Comparative Diagrams for Quantifiers

After having computed the quantifiers, we want to visually compare them in order to see if there is a correlation between them.

A good visualization of this can be seen in figure 4.6. This figure compares the spike count with the peak firing frequency in recording 21. The time is plotted on the x axis ranging from the start of the recording to the end. This recording lasted over 10,000 seconds. For each spike train we plotted the peak firing frequency, the spike count as well as the spike train duration in separate rows of the figure. In addition we added the electrical stimulation frequency, to see what effect the changes in frequency have on different quantifiers. Figure 4.7 shows the same data in a slightly different format. Here the electrical stimulation frequency is included in the rows where we plot the quantifiers to better see the immediate effect.

The first thing that we can observe is that the peak firing frequency and spike count behave very similarly. Secondly, the application of different electrical stimuli does seem to have an effect on the spike trains. We can see that during the periods with 4 and 5 Hz stimulation the peak firing frequency as well as the spike count take a significant dip. However, the same cannot be said for the interval with 2 Hz stimulation. The quantifiers stay largely the same, which leads us to the assumption that there is some sort of threshold which needs to be passed in order to influence the spike trains. In our recordings this threshold

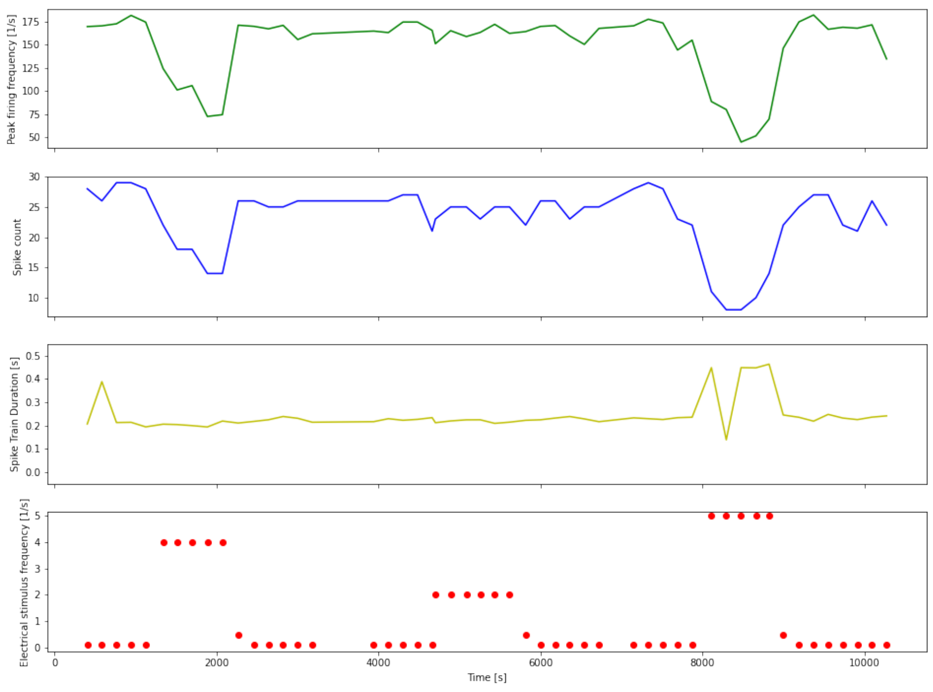


Figure 4.6: Diagram showing a separated comparison different quantifiers for recording 21. In the first row of the figure, the peak firing frequency (1/s) is plotted for each spike train in the recording. The second row shows the spike count and the third row the spike train duration (s) for the trains. The last row shows the level of electrical stimulation frequency (Hz) during each train in red dots.

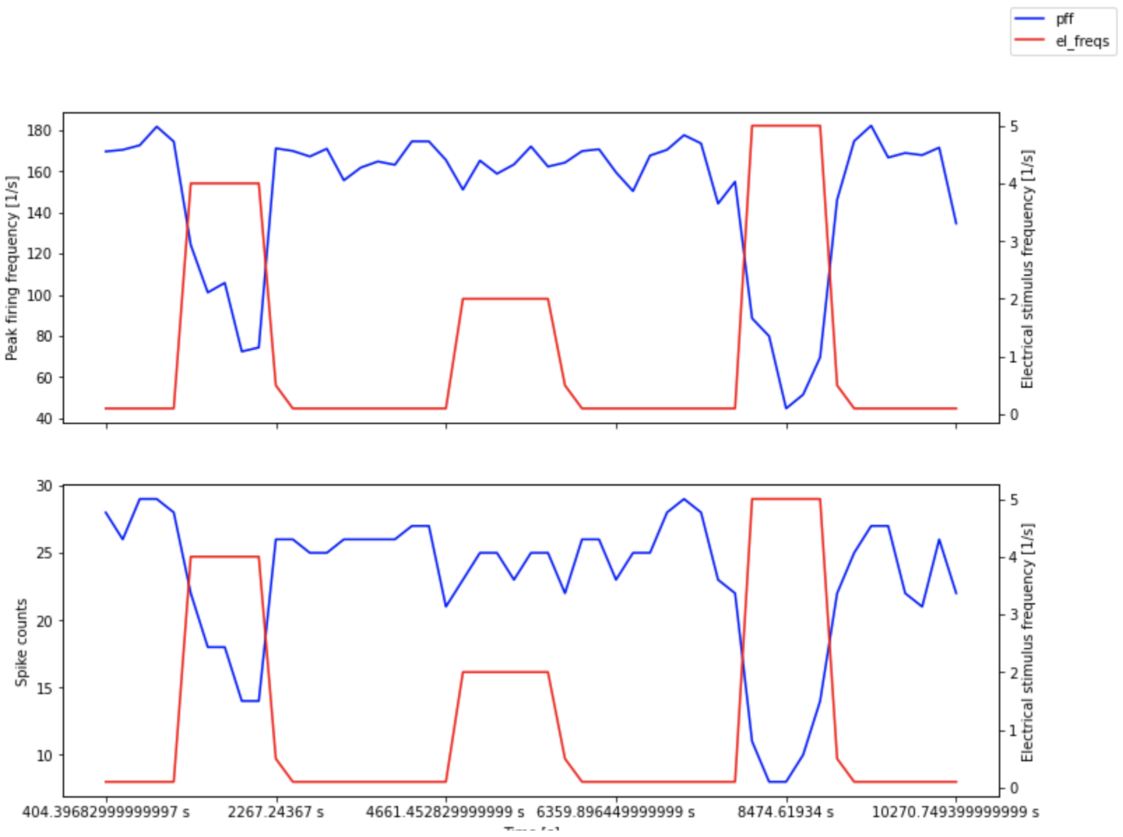


Figure 4.7: Diagram showing a comparison of peak firing frequency and number of spikes for recording 21

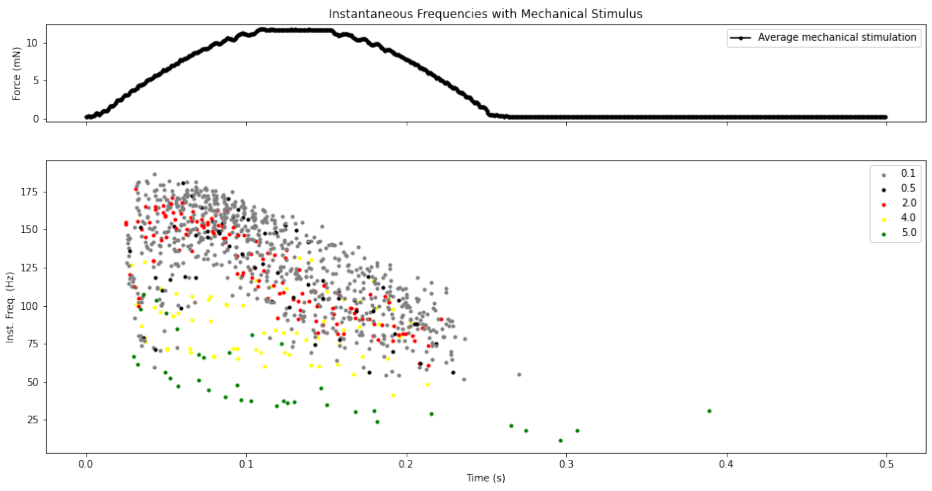


Figure 4.8: The top part of the figure shows the averaged mechanical force (mN) during stimulation for recording 21. The lower part details the instantaneous frequencies for each spike train. The instantaneous frequencies were averaged with a sliding window of three to lessen the impact of outliers. Spikes occurring during 0.1 Hz stimulation are color-coded in grey, spikes during 2, 4, 5 and 0.5 Hz stimulation are colored in red, yellow, green and black respectively.

seems to lie somewhere between the 2 and 4 Hertz electrical stimulation marks. While the peak firing frequency and spike count show significant variation, the pike train duration stays pretty constant for the 4 Hertz stimulation but does also increase during 5 Hertz stimulation. This may point to different thresholds for different aspects of a spike train.

Instantaneous Frequencies

Another quantifier that we can take a look at is the instantaneous frequencies of the spikes in the spike trains and observe how these change with different levels of electrical stimulation frequency. To visualize this we recreated a figure from the original paper from Uebner et al. in figure 4.8. The data for this figure is taken from recording 21. The x axis depicts the time since the last stimulus in seconds. The top part of the figure shows the force of mechanical stimulation that evoke the spike trains. This curve represents the average mechanical stimulus in this recording. A sliding window of three was used for the calculation of the instantaneous frequencies to smooth out some of the outliers. The lower part of the figure is a scatterplot of these instantaneous frequencies for every spike train in the recording. They were color-coded according to the underlying electrical stimulation frequency. Spike trains with a frequency of 0.1 Hz are colored in gray, while trains with higher stimulation are colored in red, yellow and green for an electrical frequency of 2, 4 and 5 Hz respectively. Finally, trains with a stimulation of 0.5 Hz stimulation frequency are colored in black. From the scattering of the dots we can observe that the spike trains occurring during an electrical stimulation of 4 and 5 Hz show lower instantaneous frequencies than the base level spike trains. The red dots, belonging to the 2 Hz stimulation, lie mostly in the main cluster of dots belonging to the base level spike trains, which would fit with the previous figures and the assumption of some sort of threshold for spike train alterations.

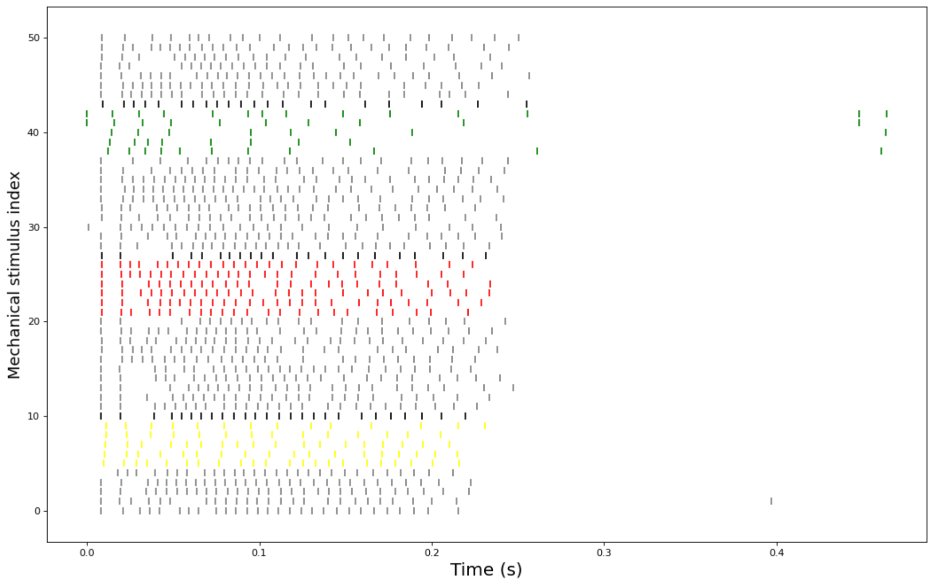


Figure 4.9: Color coded event plot. Every line in this plot represents a spike in the recording. The time since the latest mechanical stimulus is given on the x axis. The y axis differentiates the spike trains, which are numbered in order of appearance. Spikes are colored grey, red, yellow, green, black when occurring during electrical stimulation with a frequency of 0.1, 2, 4, 5, 0.5 Hz respectively.

Event plot

A figure that we have seen previously in this chapter is the event plot. It shows an overview of spiking activity after mechanical stimuli over the course of a whole recording. Figure 4.9 shows a color coded event plot. This shows the same recording as figure 4.8 and uses the same color coding for the electrical frequencies. It is not meant to be an in depth analysis tool, but rather a quick visual representation of a recording. This can give us an idea if it makes sense to further analyze a particular recording, by looking at the length of the spike trains.

Chapter 5

Discussion

5.1 Software

During the course of this thesis I have worked with multiple versions of the openMNGlab framework. The biggest issue with the software at this point are the importing capabilities. This is a very crucial functionality necessary for everyone who works with the software. Other users have reported that some aspects can not properly by imported into the framework such as the comments in chemical experiments. This particular issue might not be necessarily on the framework itself but on how the experimental protocol is handled by the experimenters. But there are other cases where the functionality can be improved. In my case I also had difficulty importing everything that was needed for the analysis. With the current Neo importer and the type of experiment in this thesis, there is mechanical information that gets lost, when just using the current framework version. I suspect that this is something that can be rectified and that the import of the mechanical force channel, as well as the missing wavemark channels might be possible in the future.
Another topic worth discussing is the possibility of storing the data and having the option of reloading it at a later time. This would lead to faster loading times when revisiting already analyzed data. Neo supports this by providing the option to save Neo objects to the hdf5 file format. This could be used for the thesis data only if everything can be imported by the Neo importer and put into one single data structure, which is another reason to look into this in the future.
Since every experiment produces different data with different specifications, there might come a point in the future, where new issues with new data sets arise.

5.2 Spike Analysis

In the results chapter we have seen what effect continuous nerve fiber activity can have on the response to a new stimulus. With increased fiber activity, induced by electrical stimulation, the spike train elicited by a mechanical stimulus changed in comparison to the control. The higher the electrical stimulation becomes, and with it the fiber activity, the fewer spikes are produced by the mechanical stimulus. This relation is not linear, however. There seems to be a threshold of electrical stimulation frequency after which we notice a change in response to the mechanical stimuli. In figures 4.6, 4.7, 4.8 we see that an electrical background stimulation frequency of up to 2 Hz does not seem to have any

effect on the peak firing frequency or the number of spikes per train. With the 4 and 5 Hz stimulation, however, we see a noticeable dip in peak firing frequency and number of spikes per train. This would suggest that a high load on the nerve fiber hinders the regular response to the mechanical stimulus that we see without much activity on the nerve fiber. This corresponds well with the results found by Uebner et al. [9].
In this thesis the detection of spike trains is very basic. We get the start time by the event marker in the recording file and then define the spike train as all the spikes that appear in a period of up to 500 ms after that event. For the purposes of these experiments this is enough, since we will always know when the spike trains are supposed to happen. When this sort of analysis gets expanded to also search for random spike trains in patients some time in the future, there needs to be more sophisticated ways to detect spike trains when there is no way to know exactly when one will happen.
Another interesting aspect for the future is the integration of Elephant into the existing analysis. Since Elephant is also a Python module and implements advanced analysis functions for spike trains it might serve as a good addition.
In this thesis we focused mostly on the analysis of single recordings from mechanically and electrically stimulated rats. This could be expanded to more big picture statistics comparing multiple different recordings in a broader context than in this thesis, as well as comparing differing stimulation types.

Bibliography

[1] "Iasp website \http://www.iasp-pain.org/," July 2022.

[2] D. Bouhassira, M. Lantéri-Minet, N. Attal, B. Laurent, and C. Touboul, "Prevalence of chronic pain with neuropathic characteristics in the general population," *PAIN*, vol. 136, pp. 380–387, June 2008.

[3] O. van Hecke, S. K. Austin, R. A. Khan, B. H. Smith, and N. Torrance, "Neuropathic pain in the general population: A systematic review of epidemiological studies," *PAIN®*, vol. 155, pp. 654–662, Apr. 2014.

[4] K. G. Brooks and T. L. Kessler, "Treatments for neuropathic pain," *Clin. Pharm*, vol. 9, no. 12, pp. 1–12, 2017.

[5] F. Rieke, D. Warland, R. d. R. Van Steveninck, and W. Bialek, *Spikes: exploring the neural code.* MIT press, 1999.

[6] P. Dayan and L. F. Abbott, "THEORETICAL NEUROSCIENCE : Computational and Mathematical Modeling of Neural Systems," p. 432.

[7] B. Namer and H. O. Handwerker, "Translational nociceptor research as guide to human pain perceptions and pathophysiology," *Experimental brain research*, vol. 196, no. 1, pp. 163–172, 2009.

[8] F. Schlebusch, F. Kehrein, R. Röhrig, B. Namer, and E. Kutafina, "openMNGlab: Data Analysis Framework for Microneurography - A Technical Report," *Studies in Health Technology and Informatics*, vol. 283, pp. 165–171, Sept. 2021.

[9] M. Uebner, R. W. Carr, K. Messlinger, and R. De Col, "Activity-dependent sensory signal processing in mechanically responsive slowly conducting meningeal afferents," *Journal of Neurophysiology*, vol. 112, pp. 3077–3085, Sept. 2014. Publisher: American Physiological Society.

[10] "CED Spike2: System introduction."

[11] "DAPSYS Data Acquisition Processor System."

[12] "R&D Research & Development."

[13] J. H. Siegle, A. C. López, Y. A. Patel, K. Abramov, S. Ohayon, and J. Voigts, "Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology," *Journal of Neural Engineering*, vol. 14, p. 045003, Aug. 2017.

[14] R. Oostenveld, P. Fries, E. Maris, and J.-M. Schoffelen, “FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data,” *Computational Intelligence and Neuroscience*, vol. 2011, p. e156869, Dec. 2010. Publisher: Hindawi.

[15] M. Denker, A. Yegenoglu, and S. Grün, “Collaborative HPC-enabled workflows on the HBP Collaboratory using the Elephant framework,” in *Neuroinformatics 2018*, p. P19, 2018.

[16] G. S., G. D., J. F., J. T.R., P. R., R. P.L., R. C., S. A., W. T., Y. P., and D. A.P., “Neo: an object model for handling electrophysiology data in multiple formats,” *Frontiers in Neuroinformatics*, vol. 8:10, February 2014.

Appendix A

z. B. Programmdokumentation