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. A. Morrison

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

Advisor: Dr. rer. medic. E. Kutafina

The present work was submitted to the Chair of Software Engineering

Aachen, June 23, 2022

Kurzfassung

Eine kurze Zusammenfassung der Arbeit.

Abstract

A short abstract of this thesis.

Contents

1 Introduction **1**

2 Background **3**

2.1 Data acquisition . . . 3

2.1.1 Spike2 . . . 3

2.1.2 Dapsys . . . 5

2.1.3 OpenEphys . . . 5

2.2 Common Spike Train Analysis Methods . . . 7

2.3 Spike Train Analysis Tools . . . 7

2.3.1 FieldTrip . . . 7

2.3.2 Elephant . . . 8

2.4 openMNGlab . . . 8

3 Methods **11**

3.1 Software . . . 11

3.1.1 Data structure . . . 11

3.1.2 Development Process . . . 13

3.1.3 Use Cases . . . 14

3.2 Spike Analysis . . . 16

3.2.1 Data . . . 16

3.2.2 Spike Train Definition . . . 17

3.2.3 Spike Train Analysis: Quantifiers . . . 17

4 Results **19**

4.1 Software . . . 19

4.1.1 Finished analysis pipeline . . . 19

4.2 Spike Analysis . . . 20

4.2.1 Experimental Protocol . . . 20

4.2.2 Sample Analysis . . . 21

5 Discussion **27**

5.1 Future Work 27

6 Conclusion **29**

References **31**

A z. B. Programmdokumentation **33**

Chapter 1

Introduction

In this chapter I will give an introduction to the topic. I will give background information about neuropathic pain and the big picture goal of research in this field.
-quickly explain how the transmission of pain functions inside our bodies
-what is my bachelor thesis based on (paper from Roberto)
-contextualize thesis topic within the big picture
-talk quickly about openMNGlab and goal of adding analysis functionality and discuss software engineering goals for the framework

Chapter 2

Background

In this chapter I will present relevant work in this field, especially for my analysis.
-Data acquisition systems: Spike2, Dapsys, OpenEphys
-Software analysis tools FieldTrip, Elephant, openMNGlab
-Neo package (included in section about openMNGlab)
-possible inclusion and cooperation with other software, such as elephant in openMNGlab context

-

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. These systems are used whenever researchers want to record electrophysiological data. There are three systems that I will describe here and that should be compatible with OpenMNGlab. These systems are Spike2, Dapsys and OpenEphys.

2.1.1 Spike2

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

The software can record multiple channels simultaneously. An example screenshot from 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 mechanoelectrostimulator applying electrical and mechanical stimulation.

First of all it contains a channel for the recorded raw signal at the bottom. The next channel contains the temperature during the recording. 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 there are spikes in mechanical force whenever a mechanical stimulation occurs to evoke a spike train. For this experiment we want to

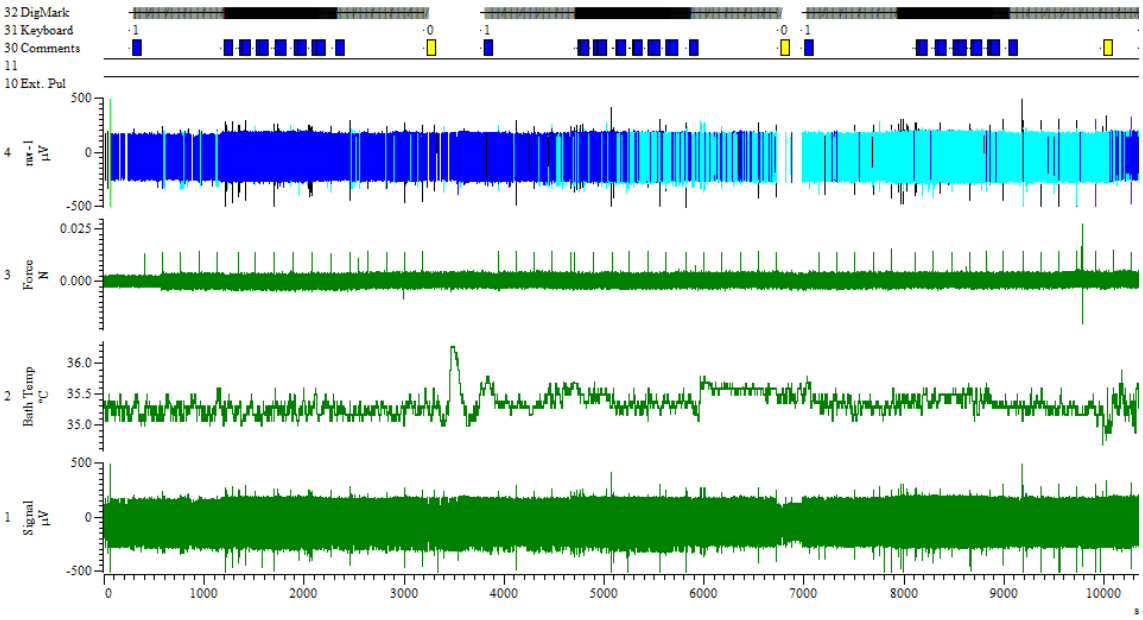

Figure 2.1: Typical mechanically and electrically stimulated recording in Spike2

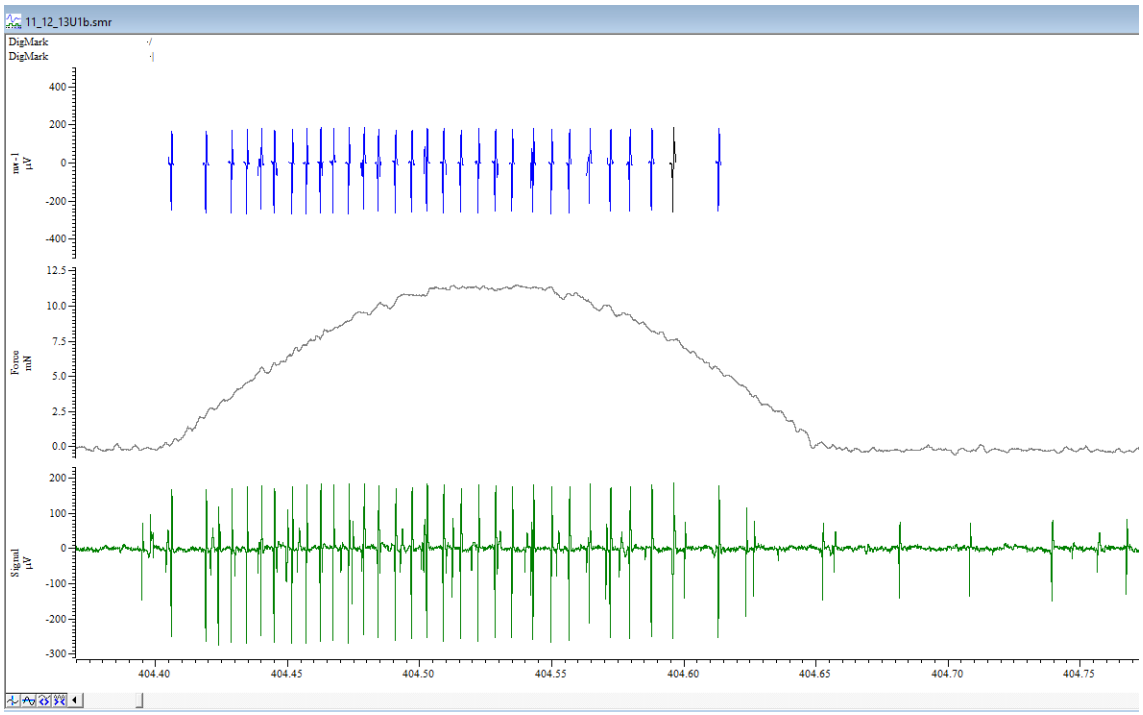

Figure 2.2: A single spike train in spike2

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 filter out specific fibers. These filtered fibers are then displayed in so called wavemark channels. In this example channel 4 is such a channel, where only specific action potentials are filtered. The filtering process is done by the experimenters and is based on certain features of the action potential shape.
The topmost channel in Figure 2.1 contains markers for the electrical and mechanical stimuli. Additionally there is a channel containing comments regarding the experiment. Comments can represent the experimental protocol and are filled in by the experimenters. In this example there are comments denoting a change 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 can be seen in Figure 2.2. Here the difference in electrical and mechanical event markers in the topmost channel can be seen. 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 [**?**]. 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 [dap].
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 is used especially for electrophysiological recordings with human patients. As is the case for Spike2 it also comes with a visual representation of the data which can be seen in Figure 2.3. This graphical interface works real time while recording the data.
The version of Dapsys used in the experiments from Barbara Namer is specialized for microneurography and was configured in cooperation with Brian Turnquist.

2.1.3 OpenEphys

The last data acquisition system I want to focus on is called OpenEphys [SLP+17]. 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.

The idea behind OpenEphys came after an increased popularity of closed-loop experiments in neuroscience, in which the results of the recorded system has an influence on the system itself. With proprietary systems it is somewhat difficult to share the details of

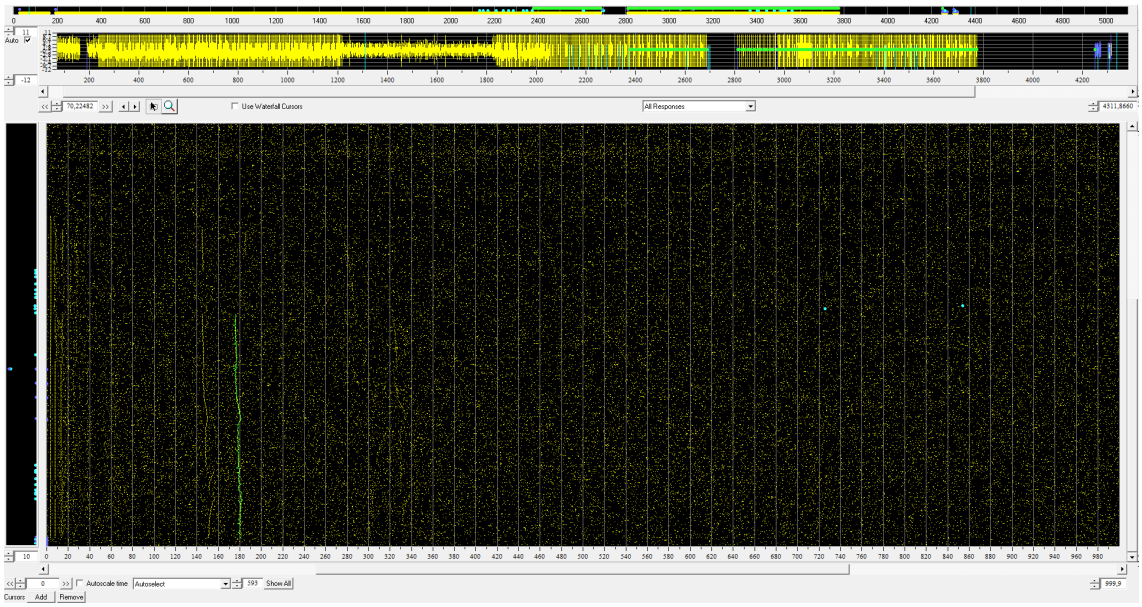

Figure 2.3: Screenshot of a recording in Dapsys

such experiments and to replicate them. The introduction of OpenEphys, an open-source system is supposed to make it easier to develop and share analysis and details of such closed-loop experiments.
OpenEphys makes use of inexpensive open-source hardware such as Intan chips [SLP+17] to make it easier for small labs to get started with analysing 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 analysing 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 come with very 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 develop new modules by making use of the class inheritance capabilities of C++.
OpenEphys comes with 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 also 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 a collaborating research team in Bristol, who should also be able to incorporate OpenMNGlab in their analysis.

2.2 Common Spike Train Analysis Methods

Here I will describe the quantifiers that other people are using such as histograms. Also describe what quantifiers are usually used to decribe spike trains.

I will now describe the most common methods people apply when analysing spike trains.

The type of diagram that is found most often in the literature is the histogram.

2.3 Spike Train Analysis Tools

I will now present two software tools for microneurography analysis, before focusing on OpenMNGlab.

2.3.1 FieldTrip

Fieldtrip [OFMS10] is a MATLAB toolbox developed at the Radboud University, Nijmegen, the Netherlands and offers a wide variety of analysis functions. It can analyse 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 somewhat 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 [OFMS10]. The first step after loading the data set of interest is to choose a data segment that is to be analyzed by setting the boundaries of said 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-using 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 (https://elephant.readthedocs.io/en/latest/index.html) 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. We want to start with a very basic look at spike trains, which is something that the Elephant package does not provide. For the use in this thesis I 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. 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 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 software systems, put them in a compatible format and analyse 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 the Institute of medical informatics has started to develop their own software framework in python called openMNGlab [SKR+21]. 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 provides analysis capabilities for microneurography data.
OpenMNGlab was started as a project of the Institute of medical informatics 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 big picture 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 necessarily usable for every recording that should have been analysed.

To help with these issues, Neo(neuralensemble.org/neo), 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 is already developed with the help of the Neo package and therefore many experiments that were recorded with the Spike2 software can be analysed using 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. We need to export the raw data we want to analyse as csv files from the Dapsys software, before we 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 we would like to be able to analyse.

The Matlab functions offered by FieldTrip, while similar from a functionality point of view are not very 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.

Chapter 3

Methods

In this chapter I will describe the methods I used for creating the software and spike analysis results in this thesis
-Describe the different data structures (Neo, old openMNGlab version)
-Detail the development process, issues during development and how they were resolved
-Describe the finished analysis pipeline with the help of a simplified graph
-present use cases (3 students analysing different data, experimental scientist)

Spike Analysis:
-Data
-definition of spike trains
-quantifiers

3.1 Software

3.1.1 Data structure

In this section I will present the details of the different data structures that I will use in the course of my analysis pipeline.

Neo structure

Neo models electrophysiological data in a hierarchical structure, which is depicted in Figure 3.1. On the lowermost level we start with 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, as the name suggests. A *SpikeTrain* object contains time point data with the information when action potentials occur. These *SpikeTrain* objects are, however, slightly different from our understanding of spike trains in this thesis. We understand a spike train as the spikes that occur resulting from one stimulus and is quite limited duration wise. The *SpikeTrain* object in the Neo structure is a collection of all spikes during one recording and all spike trains as we see them would be contained in one such *SpikeTrain* object.

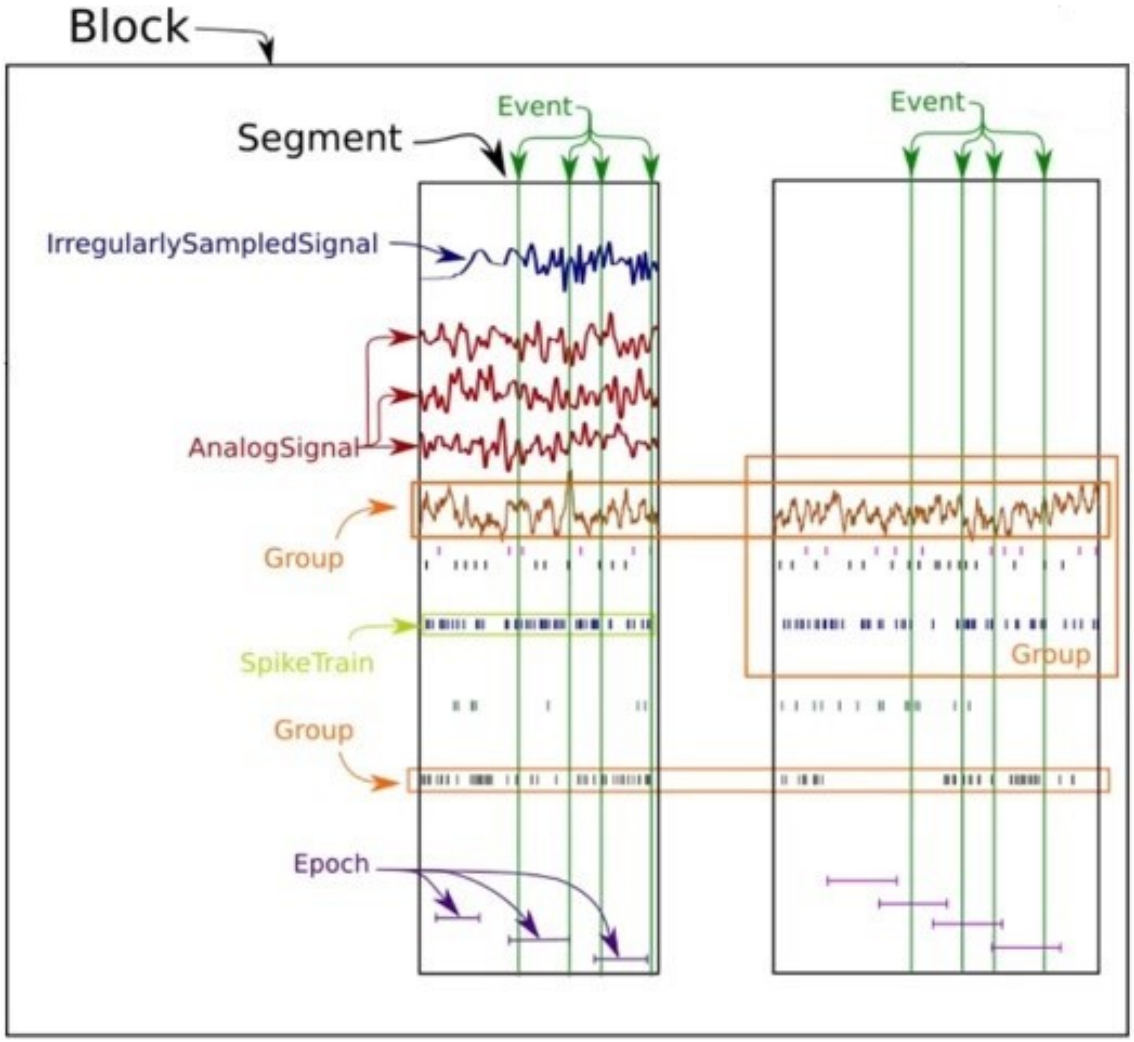

Figure 3.1: Structure of Neos hierarchical model for electrophysiological data.

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 objects for grouping all of these lower level objects. The first of these objects is a *Segment*. This groups data that was simultaneously recorded. Then there are so called *Groups* which can group data in any arbitrary way and is not restricted to the data being in the same Segment. On the topmost layer there are *Blocks*. One *Block* contains multiple *Segments* and *Groups* which in turn 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

Part of my analysis is still using code from version 1 from openMNGlab where the Neo structure was not yet integrated. The importer from that version delivered the data in a custom data structure.
At its core the custom data structure models the data hierarchically, but the objects look a little different from 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.

3.1.2 Development Process

The basis of my 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. From this he extracted the spike trains and created figures such as event plots as seen in Figure3.2. These event plots depict the spike trains of a single recording. Each row contains the time point data of spikes for 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.
Starting of with this way of handling the data, I continued my own analysis by using the importers from version 1 of openMNGlab. Taking the extracted information of mechanically induced spike trains I started with quantifying these trains with simple numbers like duration or number of spikes per train.
With the further development of openMNGlab and the addition of the Neo package the importing process for the Spike2 files became a lot easier and faster. However, after a couple of attempts it became clear that the importer in its current state did not import the mechanical force channel from the experimental files. This might not be an issue when analysing other types of data, but for our use case it presents a big issue. Because the information of the mechanical force is required for the analysis I chose to combine parts of the old version of openMNGlab and parts of the new version. This way there is benefit from using the structure that Neo brings when it comes to speed and ease of use, but I am not missing information for the analysis.

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

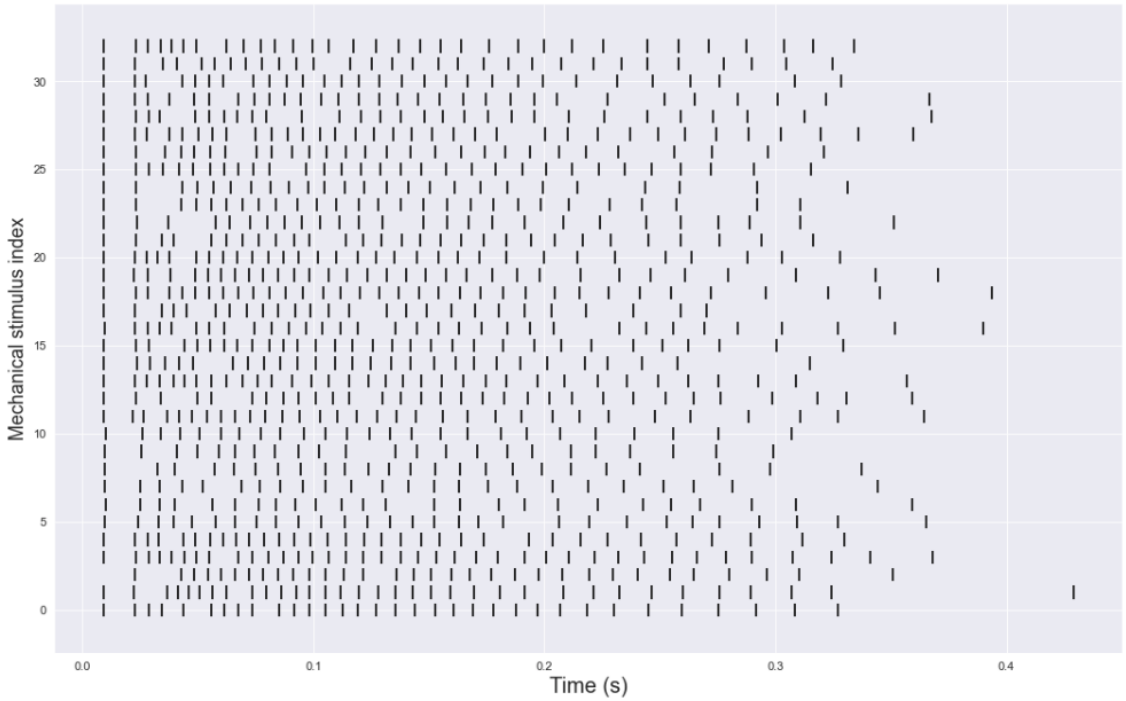

Figure 3.2: Event plot detailing spiking activity for one recording.

in many cases. In the files that I was provided 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. I circumvented this problem by deleting the comment channels in those files, as they did not contain relevant information for my analysis. In other cases this might not be a possible solution and either the experimenters have to agree to not use incompatible symbols in their comments or the importer gets extended to also recognize greek letters.

3.1.3 Use Cases

To end this section I want to present a couple different use cases of various users of openMNGlab. I will describe the specifics of each use case and then give a few remarks on the 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 a couple of 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 already we encounter some 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 analyse 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 framework was working properly and therefore uses their own custom analysis notebook. However, all the information that is required is also available with the regular Spike2 importer in the current openMNGlab version. 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 needs to be addressed if we want to work with recordings that feature multiple types of stimulation. Additionally there might be some general import difficulties which I will elaborate on later in this section.

Student 2

The second user is also a student who uses mechanically and electrically stimulated Spike2 data. Their goal is to analyse spike trains resulting from mechanical stimulation. For this they do not need all of the information contained in the raw Spike2 file. They need the event information for mechanical and electrical stimuli as well as the mechanical force information for details of the mechanical stimuli. The event information can be found in the DigMark channel of the Spike2 file and gets extracted by the Spike2 importer of openMNGlab. The mechanical force is a continual signal but appears in the form of sin shapes when the nerve is stimulated. Just as the first user they also need the information regarding the spikes themselves, which are also imported by the framework.
From the gathered data the user then calculates various quantifiers for the mechanically induced spike trains such as spike count or more elaborate features. The user started this project while still using the old framework version where the channels from the Spike2 software were manually extracted to a csv file. With the updates to the framework in the meantime not all of the required information gets extracted as easily as before. This is why they use a hybrid version of the two versions of the framework where they mix and match the functions as they need them.

Potential problems: There are similar potential problems when it comes to the event channels and regular spike channels as in the case of the first user. Additionally there are problems when it comes to the import of the mechanical force. The user started the analysis with an old version of the framework which used other importers (csv files). With this technique they could extract the mechanical force throughout the recordings. The new Neo importer still has some bugs when it comes to importing the mechanical force from the Spike2 file, which requires fixing for future users who need this information.

Student 3

The third student user of the framework works on a project to analyse Spike2 data in which certain chemicals where 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 they only made use of the new importing functions and are not stuck with certain parts from the old framework. In addition to the spiking activity, as in the previous use-cases, they also need information regarding the chemicals that are applied. They need the timings and doses of the specific chemicals. From this they need to specify a time frame where they want 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 general comment channel in Spike2. That means that the experimenter manually creates a comment every time a new chemical is applied. The issue with this is that openMNGlab does not import the comment channel. In practice the comments do not have a strict naming scheme, which would also lead to difficulties with the automatic detection of specifics regarding the chemicals. This is a problem however that does not stem from the framework, but is something that has to be addressed on the side of the experimental researchers.
These described problems lead to the current workflow of manually selecting a suitable time frame for interesting spiking activity some time after the chemical application; some times up to a minute (check the times again) after the chemicals where applied.

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.

3.2 Spike Analysis

In this section I will describe the methods I used for analysing the spike data.

3.2.1 Data

Because human nerve data is hard to obtain, we can also use animal data instead as a proxy. Animal data is usable as 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 (put reference). The goal of the paper was to evaluate the effects of spiking activity on the 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 sinusoidal shape.
For single recordings the mechanical force that is applied throughout stays at approximately the same level for most of the files (put for how many files this is the case), but there are exceptions where the mechanical stimulation changes in amplitude and length during one recording.

3.2.2 Spike Train Definition

This subsection should contain the definition we use for defining spike trains. If we need a better understanding of spike trains in the Software section this subsection might move to the background chapter.

Earlier in this thesis we covered the basics of electrical signals in nerve fibers and action potentials, or spikes. 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 interestet in is the short amount of time after a stimulus event where there is increased spiking activity. In the data provided by Roberto de Col there is, as described in the previous section, there is mechanical stimulation. This mechanical stimulation, which is approximately 250 milliseconds long, leads to noticably increased spiking activity. It is this spiking activity that we want to analyse 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 increaded spiking activity has normalised. This can have a duration ranging from 150 milliseconds to 500 milliseconds. Due to computational reasons in our spike train detection we are not using any complicated algorithms but only search an interval of 500 milliseconds after the mechanical simulus to look for spikes for the spike trains. Any spikes happening after that interval do not get counted with our current method. All of the spikes happening inside of this interval get counted as belonging to the spike train.

3.2.3 Spike Train Analysis: Quantifiers

I will present the quantifiers that I chose to use and explain the reasoning.

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 quite simple quantifier I found 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 I chose to include is the peak firing frequency(pff). The reason for the inclusion is because it is one of the quantifiers mentioned in Roberto de Cols original paper about this data set. The peak firing frequency is calculated for each spike train and details the concentration of spiking activity. During development I experimented with two different implementations of calculating the pff. First I tried sorting the instantaneous frequencies of a spike train by value and averaging over the 5 highest frequencies. This, however, does not give a representative look on a single section of spikes. Therefore switched to a sliding window implementation, where I calculated the average instantaneous frequency over a sliding window of width 5 and took the highest average as the peak firing frequency.
At the request of my advisor I calculated the inter spike intervals(ISI) for the spike trains. This can give us a measure of the distribution of spikes in a spike train.
Going off of the ISI we can compute the linearity of the inter spike intervals in the form of R2. We want to know if the ISIs tend to increase as the spike trains last longer or decrease or if there is no significant impact. From the start we suspected that the intervals increase as the spike train goes on and calculating the R2 value should give us the confirmation.

Chapter 4

Results

4.1 Software

4.1.1 Finished analysis pipeline

The software development process described in the previous chapter resulted in a jupyter notebook, which represents the current analysis pipeline used for the analysis in this thesis. A schematic version of this pipeline can be seen in figure TODO. For a more detailed view on the different functions, have a look at a more detailed diagram or the code in the appendix.

Importing data

The first part of the notebook is the import of the data, which consists of two separate importing steps. First we use the Neo importer from the current version of our framework to extract the information about the underlying electrical stimulation. After that we use the importer from the old version of openMNGlab to get the spike timings as well as the information about the mechanical stimuli. After the two importing steps we end up with two separate data structures, that we discussed in the previous chapter, 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, we receive with the second importing step. This information is contained in the custom data structure. The spike timing would also have been available in the newer Neo format, however, the analysis code was started with the custom data structure in mind, so I did not change that part of the code for the purposes of this thesis. For the future it would be a good idea to switch fully to the Neo structure, so that there is a unified structure for everything.

Preprocessing

The next part of the notebook contains internal processing steps to sort the spike trains and prepare the data for easy representation and quantification.
Making use of the early versions of OpenMNGlab and the first analysis notebooks from

Radomir the event plot for the current file gets computed and visualized. The next step is calculating the inter spike distances, meaning the time between two spikes, and creating inter spike distance graphs for each spike train in the recording.

Computing quantifiers

Now follows the main part of quantifier computation. This is where the computations of my chosen quantifiers happens. After all the quantifiers are computed the resulting lists of quantifiers for each spike train get put into a data structure and saved for later reimporting and reuse for further analysis.

Visualizations

Now that the quantifiers are calculated it is time to visualize the results. For this, I create different diagrams showing quantifiers over the whole recording and comparisons of different quantifiers. These figures will be explained in the analysis part of this chapter.

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

This subsection will explain how the recording files are structured to get a better general understanding of the experiments and analysis.

As mentioned in previous chapters, the files I am working with feature electrical and mechanical stimulation. Mechanical pressure is applied to the nerve fiber with a mechanostimulator described in [UCMDC14] in a sinosoidal shape as can be seen in Figure 2.2. The attributes of this type of stimulation, like amplitude or duration, is constant over the course of single recordings, but may change for different recordings. In addition to the mechanical stimulation there is also electrical stimulation, which is applied as electrical impulses with a certain frequency. The base frequency of the electrical stimulation is the same for each of the recordings and lies at 0.1 Hertz. This base frequency is applied during the whole recording with a few exceptions. Each recording features segments where there is electrical stimulation with increased frequency, during which a varying number of spike trains take place (usually $5 - 6$). After each of those segments with increased electrical frequency there is a smaller period of stimulation where the frequency is at 0.5 Hertz before it goes back to the base frequency. One spike train get triggered during each of these small periods. In the recordings that where analyzed in this thesis there are 3 different

File number	Number of trains	Avg. spikes per train	Avg. train duration
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 a little overview over the available recording files for this thesis.

levels of increased electrical stimulation frequency, which lie at $2.0, 4.0$ and 5.0 Hertz. Each file features at least one burst of increased frequency at one of the mentioned levels. The files in Table 4.1 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 $2.0, 4.0, 5.0$ 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 I will use recording 21 as an example and demonstrate the outcomes. This will give a good look at the quantifiers that I chose to utilize as well as present possible visualizations using those quantifiers.

Table of values

When applying the analysis notebook to a recording, the first basic output we get is a big table containing the finished quantifiers as well as the original timestamp values for the spikes. A sample table can be seen in Table 4.1, which shows a screenshot of the output table for recording 21. First of all the table contains the mechanical stimulus information for each train. This includes the amplitude, duration and timestamp of the

spike train	0	1	2	3	4	5	6	7	8	9	10 ...
amplitude	11,499	11,643	11,643	11,676	11,807	11,288	11,367	11,357	11,599	11,543	11,453 ...
onset	404,42425	584,6745	764,93065	945,183	1125,4322	1340,5914	1513,49405	1697,90665	1886,8237	2072,4864	2267,27155 ...
duration	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	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	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	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	0,1	0,1	0,1	0,1	0,1	4	4	4	4	4	0,5 ...
sliding window peak frequency	169,499781	170,293569	172,500197	181,561806	174,296439	124,149987	101,075938	105,767344	72,4582581	74,3865025	171,005654 ...
peak frequency (5 max)	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	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	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	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 ...
log(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	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 4.1: Sample picture of table after successful analysis

corresponding stimulus. In the figure this information is contained in the upper third of the table. In addition the table features the single value quantifiers for the spike trains that I described in the previous chapter. These include among others 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 some of the raw data such as the spike timings as well as some intermediate results such as the inter pike intervals, which can be used to compute some of the single value quantifiers. These are depicted in the lower third of the figure in an abbreviated version for the sake of visibility.

Event plot

A figure that we have seen previously in this thesis is the event plot 3.2. It shows an overview of spiking activity after mechanical stimuli over the course of a whole recording. 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.

Comparative diagrams for quantifiers

Having computed the quantifiers is all well and good, but now we want to visually compare them in order to see if there is correlation between them in any way.

A good visualization of this can be seen in figure 4.2. 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 10000 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 4.3 shows the same data in a slightly different format. Here the electrical stimulation frequency is included in the rows where we plot the quantifiers

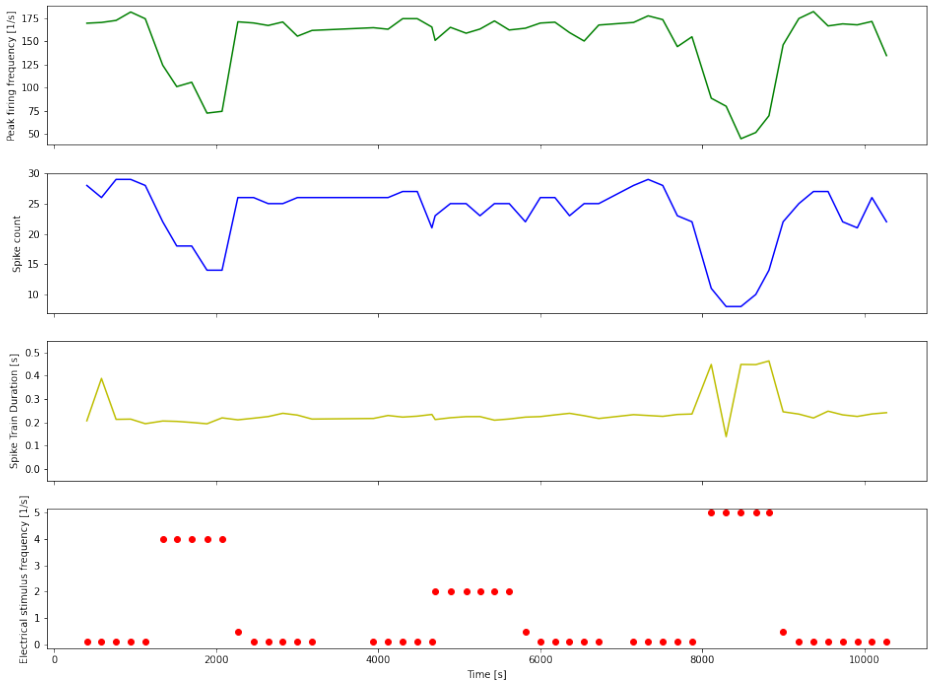

Figure 4.2: Diagram showing a separated comparison of peak firing frequency and spike count for recording 21

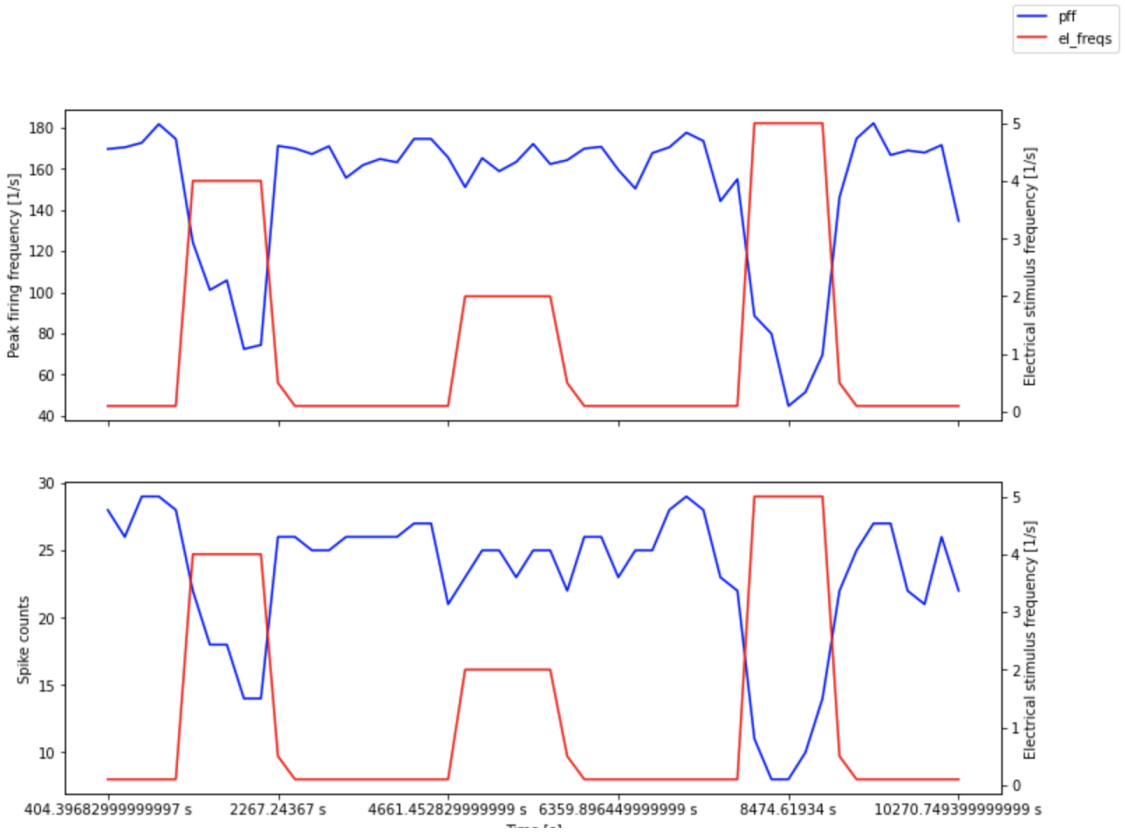

Figure 4.3: Diagram showing a comparison of peak firing frequency and number of spikes for recording 21

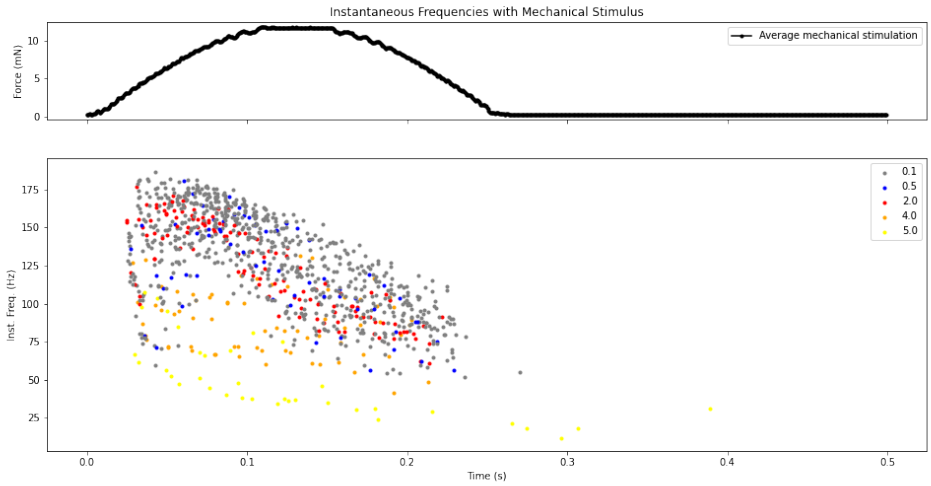

Figure 4.4: Figure showing the instantaneous frequencies in each spike train for one recording.

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 and have almost the exact same graph. Secondly, the application of different electrical stimuli does seem to have an effect on the spike trains. We can see that in the areas with 4 and 5 Hertz 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 Hertz 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 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. To visualize this we recreated a figure from the original paper from Roberto de Col in Figure 4.4. 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. The pike trains with the base level frequency of 0.1 Hertz are colored in gray, while the trains with higher stimulation are colored in red, orange and yellow for an

electrical frequency of 2, 4 and 5 Hertz respectively. Finally the trains with a stimulation of 0.5 Hertz stimulation frequency are colored in blue. From the scattering of the dots we can observe that the spike trains occurring during an electrical stimulation of 4 and 5 Hertz show lower instantaneous frequencies than the base level spike trains. The red dots 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.

Chapter 5

Discussion

In this chapter I will discuss the results presented in the previous chapter and think about possible future work regarding the analysis. Also I will discuss the integration of my code (quantifiers mainly) into openMNGlab and thoughts about the structuring of the software.

There seems to be a threshold of electrical stimulation frequency after which we notice a change in response to the mechanical stimuli. In the diagrams where we compare different quantifiers we see that a electrical background stimulation frequency of up to 2 Hertz does not seem to have any effect on the peak firing frequency or the number of spikes per train. With the 4 and 5 Hertz 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.

discuss the software engineering of openmnglab

5.1 Future Work

Since we did not use sophisticated algorithms to detect spike trains, this would be a topic for further research. Our current method relies heavily on mechanical stimulus events to tell us where we can expect a spike train to appear. This is a luxury that is not a given in microneurography data and as such it is a problem worth looking into.
Another interesting possibility for the future is the incorporation of advanced analysis functions from the likes of Elephant into OpenMNGlab. When the structure allows for it it could yield a nice benefit to the analysis capabilities of researchers who are using OpenMNGlab.
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 analysis comparing multiple different recordings in a broader context than in this thesis, as well as comparing differing stimulation types. Other students at the chair for medical informatics are already working on recordings with other stimulation types for example.

Chapter 6

Conclusion

Figure 6.1: Das SE Logo

Bibliography

[dap] R&D Research & Development.

[OFMS10] Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data. *Computational Intelligence and Neuroscience*, 2011:e156869, December 2010. Publisher: Hindawi.

[SKR+21] Fabian Schlebusch, Frederic Kehrein, Rainer Röhrig, Barbara Namer, and Ekaterina Kutafina. openMNGlab: Data Analysis Framework for Microneurography - A Technical Report. *Studies in Health Technology and Informatics*, 283:165–171, September 2021.

[SLP+17] Joshua H Siegle, Aarón Cuevas López, Yogi A Patel, Kirill Abramov, Shay Ohayon, and Jakob Voigts. Open Ephys: an open-source, plugin-based platform for multichannel electrophysiology. *Journal of Neural Engineering*, 14(4):045003, August 2017.

[spi] CED Spike2: System introduction.

[UCMDC14] Michael Uebner, Richard W. Carr, Karl Messlinger, and Roberto De Col. Activity-dependent sensory signal processing in mechanically responsive slowly conducting meningeal afferents. *Journal of Neurophysiology*, 112(12):3077–3085, September 2014. Publisher: American Physiological Society.

Appendix A

z. B. Programmdokumentation