RWTH Aachen University
Software Engineering Group

First line of title
Second line of title
Third line of title
Forth line of title

Bachelor Thesis/Master Thesis/Seminar Paper

presented by

Surname, Prename

1st Examiner: Prof. Dr. B. Rumpe

2nd Examiner:

Advisor:

The present work was submitted to the Chair of Software Engineering

Aachen, April 7, 2022

Kurzfassung

Eine kurze Zusammenfassung der Arbeit.

Abstract

A short abstract of this thesis.

Contents

Chapter 1

Introduction

1.1 Chapter content

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

Data

2.1 Chapter content

This chapter describes the type of data we are working with.
-mechanically and electrically stimulated nerve fibers from rats
-describe reasoning for using animal data as opposed to human data
-maybe as part of introduction chapter?

Chapter 3

Background

3.1 Chapter content

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

-

3.2 Data acquisition

A data acquisition system is a combination of software and hardware components that work together in order to control inputs towards and record data from different subjects. There are many different data acquisition systems for electrophysiological data. There are three systems that we are working with and that should be compatible with our chosen software analysis framework. These systems are Spike2, Dapsys and OpenEphys.

3.2.1 Spike2

Spike2 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 **??**. 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

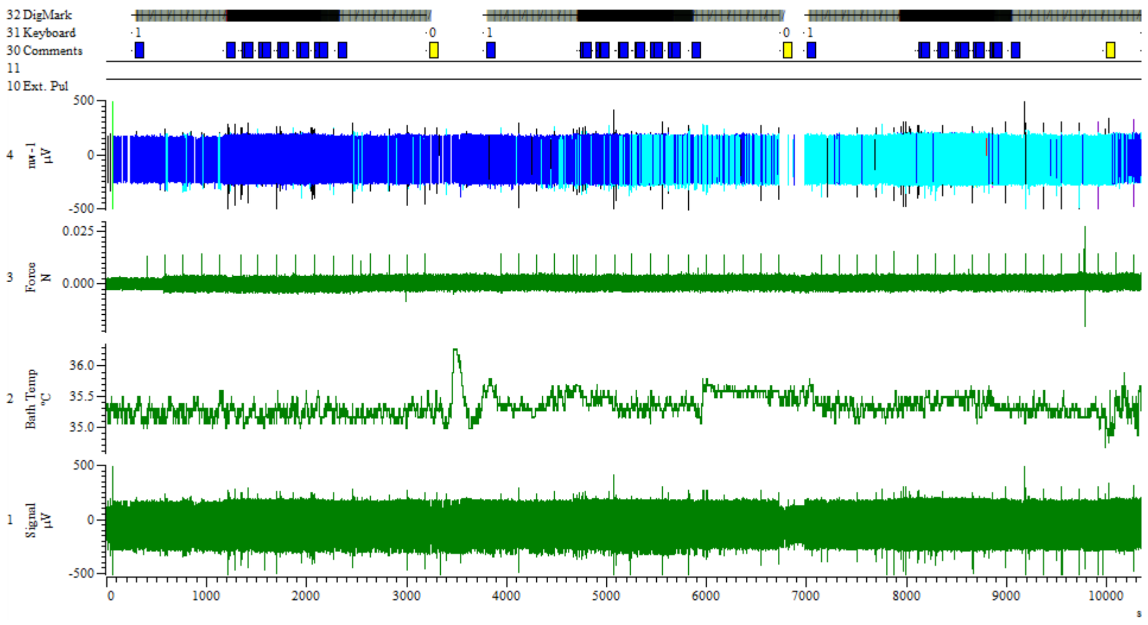


Figure 3.1: Typical mechanically and electrically stimulated recording in Spike2

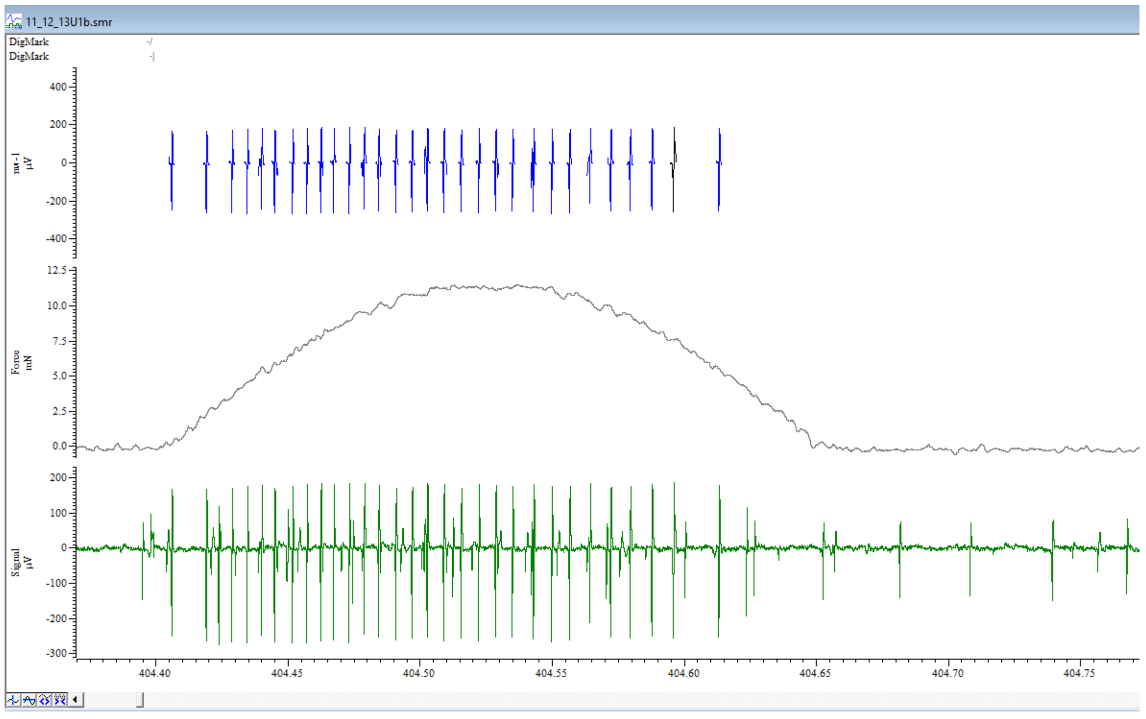


Figure 3.2: A single spike train in spike2

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 ?? there are spikes in mechanical force whenever a mechanical stimulation occurs to evoke a spike train. For this experiment we want to collect the data of single nerve fibers. It is diffucult, to record just a single nerve fiber in vitro, however. 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 ?? 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 ??. 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.

3.2.2 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 analyse electrophysiological data from animal or human sources and has been mainly used for studying the peripheral nervous system (cite website here). 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 (cite the article).
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 ??. 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.

3.2.3 OpenEphys

The last data acquisition system I want to focus on is called OpenEphys. 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.

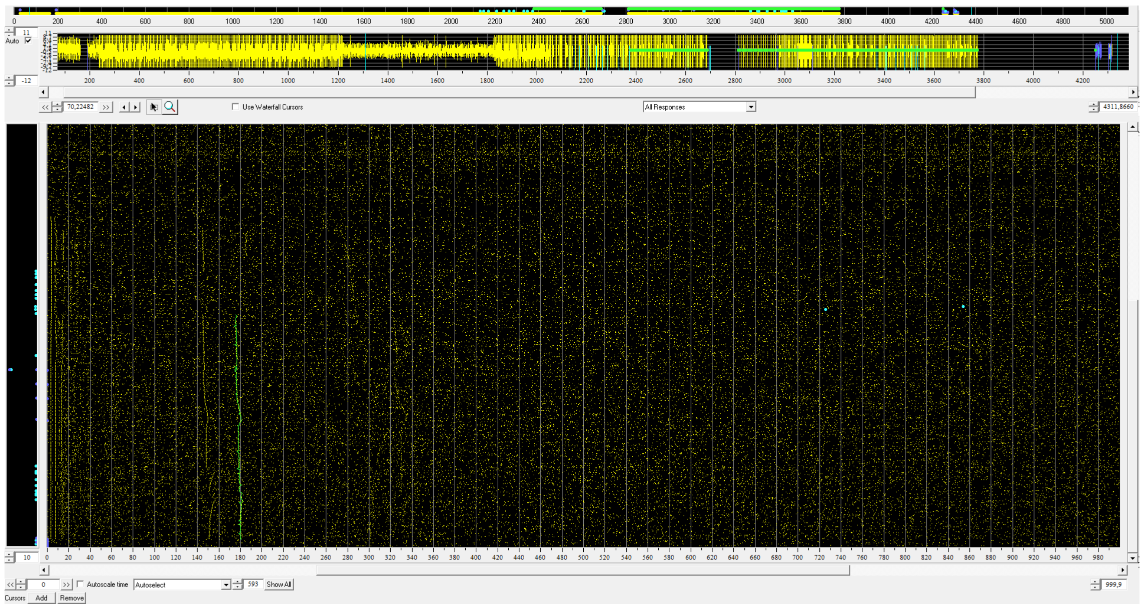


Figure 3.3: Screenshot of a recording in Dapsys

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 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 to make it easier for small labs to get started with analysing electrophysiological data. The heart of the software part of the system 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 get the system that fits the users needs exactly. 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 to 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.

In this thesis we are considering OpenEphys because it is used by a collaborating research team in Bristol, which also wants to make use of openMNGlab.

3.3 Analysis tools

When it comes to analysis software for microneurography data, there are multiple available that each have their own use-cases.

3.3.1 FieldTrip

Fieldtrip (fieldtriptoolboX.org) 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 (put citation here). The first step after loading the data set of interest is to choose a data segment that is to be analysed 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.

3.3.2 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.

3.3.3 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. 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).

3.3.4 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 4

Software

4.1 Chapter content

-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

Software engineering:
-what issues with the structure of different openMNGlab version did I encounter
-What are the needs of different users of the framework in the end?
-present use cases (3 students analysing different data, experimental scientist)

4.2 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.

4.2.1 Neo structure

Neo models electrophysiological data in a hierarchical structure, which is depicted in Figure **??**. 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.
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.

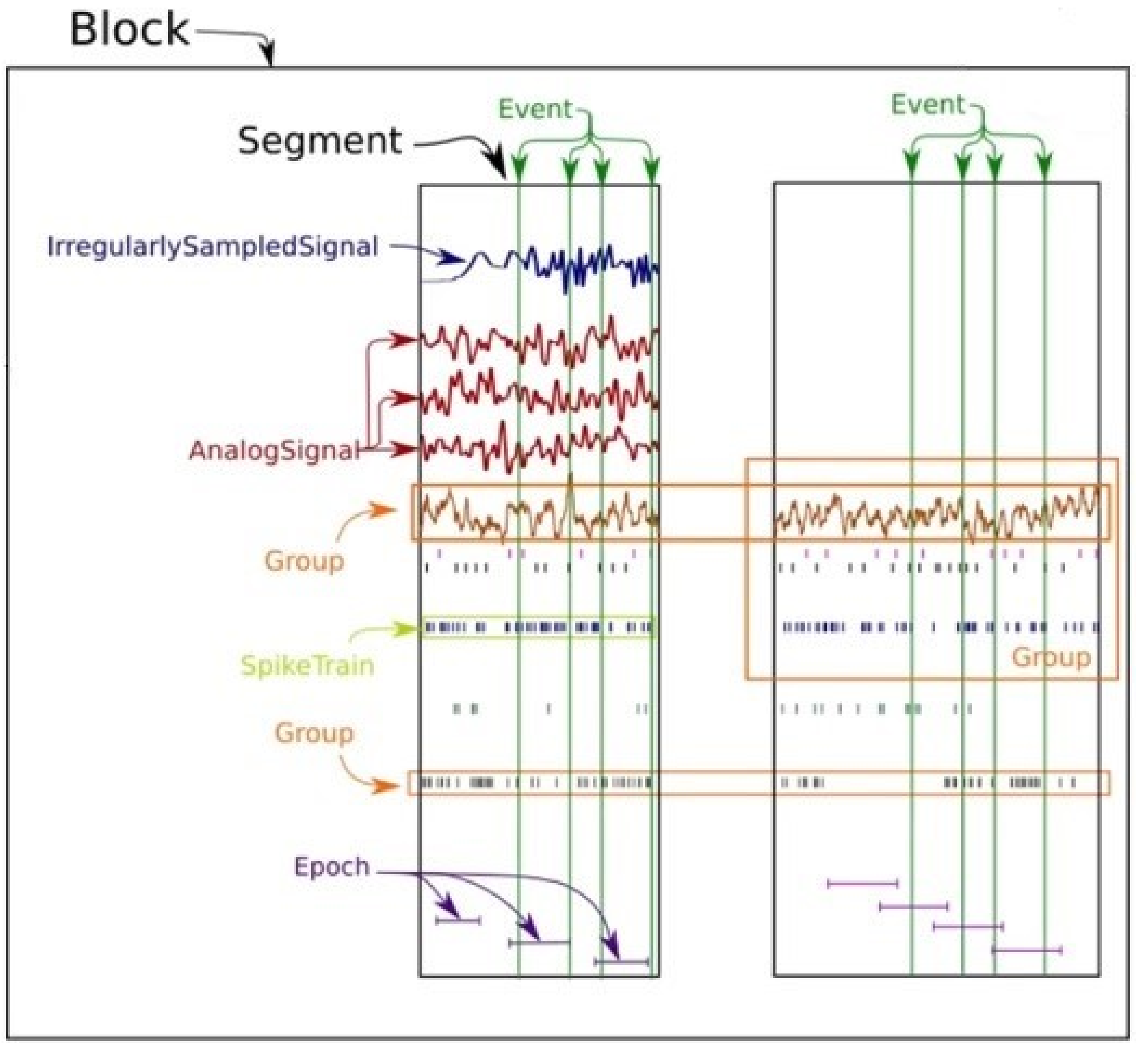


Figure 4.1: Structure of Neos hierarchical model for electrophysiological data.

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.

4.2.2 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

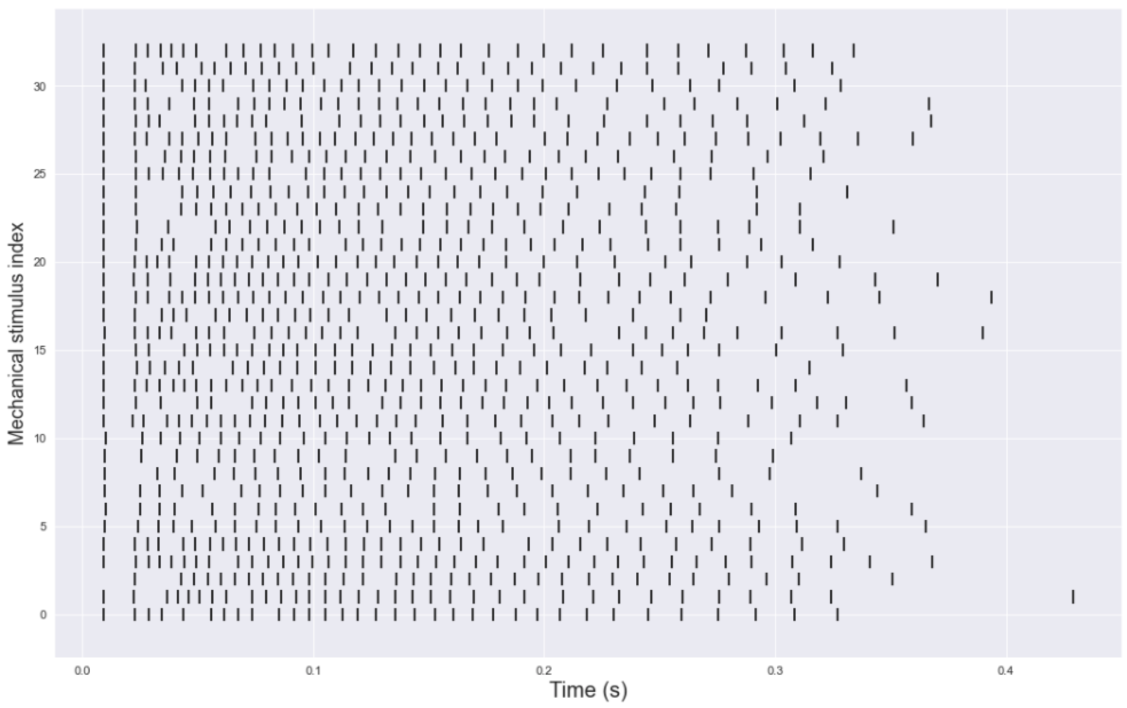


Figure 4.2: Event plot detailing spiking activity for one recording.

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.

4.3 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 Figure**??**. 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.

4.4 Finished analysis pipeline

The software development described in the previous chapter resulted in a single jupyter notebook, which represents the current analysis pipeline for this thesis. A schematic version of this pipeline can be seen in figure TODO. The import of the data 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 rest of the required information. The next part contains some internal processing steps to sort the spike trains and prepare the data for easy representation and quantification in the following steps.

4.5 Use Cases

To end this chapter 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.

4.5.1 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.

4.5.2 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.

4.5.3 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.

4.5.4 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.

Chapter 5

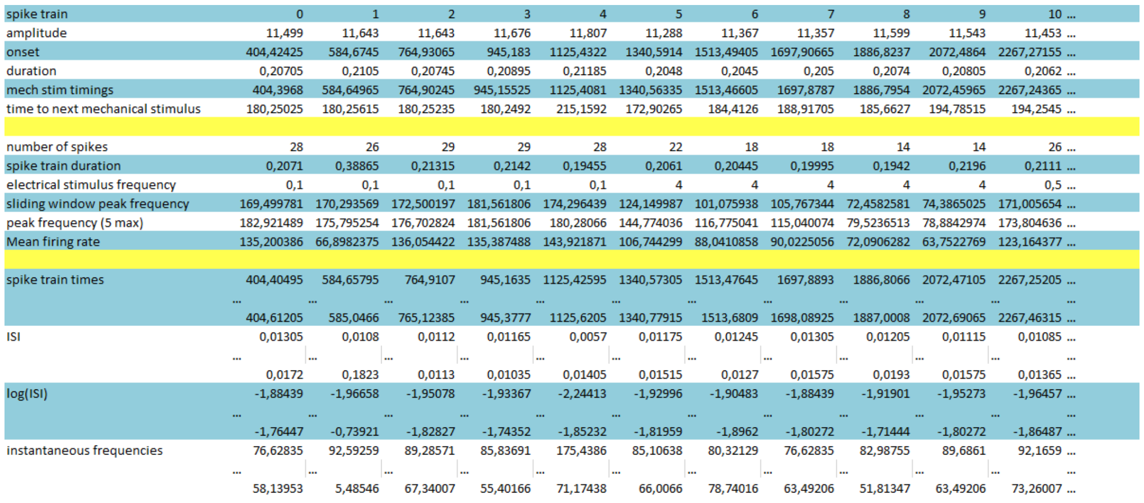
Analysis

5.1 Chapter content

In this chapter I will present the results of my analysis.
-Describe quantifiers and discuss the results of analysing the experimental recordings
-image of table containing everything the jupyter notebook has computed
-table with info on electrical frequency levels in each recording
-diagram showing a quantifier (peak firing frequency) and electrical frequency for each spike train of single recording
-compare diagrams of different recordings
-compare different quantifiers in one diagram with each other for the same file
-compare ISI to log(ISI) for every train in one file

5.2

-Show picture of a table with all the quantifiers as an example and add the rest to the appendix
-first batch contains information on the mechanical stimulus, second batch single number quantifiers about spike train, last batch lists of raw data and list quantifiers such as ISI
-with the help of this diagram describe the quantifiers and results that were computed



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 5.1: Sample picture of table after successful analysis

Chapter 6

Discussion

6.1 Chapter content

In this chapter I will descuss 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.

Chapter 7

Conclusion

Figure 7.1: Das SE Logo

Appendix A

z. B. Programmdokumentation