Imperial College
London

# Final Year Project Report
# Imperial College Bioengineering

Adrien Rapeaux

# Contents

**Abstract**

Peripheral nerve implants used to treat spinal cord injury lack selectivity, limiting their functionality. Significant research effort is ongoing to improve electrode technologies, yet some potential remains untapped in existing technologies, which can be improved by refining stimulation technique.

This project aims to improve peripheral nerve stimulation techniques using existing electrodes by modifying the stimulation protocol to obtain additional selectivity. The new protocol could be used to raise patients' quality of life and increase their independence. To accomplish this a model of myelinated peripheral nerve was created using the Frankenhaeuser and Huxley [13] equations within the NEURON [18] simulation environment. The model simulates the interactions between 2 nerve fibers of differing diameter, an internal stimulation electrode and an external blocking electrode. By timing excitatory signals and blocking signals, selective stimulation of smaller nerve fibers was obtained. Results suggest the technique could be used with existing technology in neuroprosthetics and pave the way for better treatments using Functional Electrical Stimulation. In order to facilitate further research on this subject, the model implementation in NEURON was designed as an experimental toolbox with modular components, ease of use and versatility in approaching the selective stimulation problem.

This report contains 16054 words.

# Chapter 1

# acknowledgements

I would like to thank my project supervisor Dr. Constandinou for letting me take on this project based on my own proposal and trusting me throughout with its progress.

I would like to thank Dr. Konstantin, Dr. Eftekhar and Ian Williams for their invaluable help and advice in the construction and realization of this project.

I would also like to thank Dr. Burdet for being my departmental supervisor in the Bioengineering Department.

Finally, I would like to thank Dr. Carnevale and Dr. Hines, both original developers of NEURON, for their precious advice on using the simulation environment, and for helping me find the causes to some of the more obscure and unexpected behaviours of my model. This project would not have been possible without NEURON.

# Chapter 2

# introduction

The goal of this project is to find a method to improve peripheral nerve stimulation, hereby refered to as FES (functional electrical stimulation) with modified stimulation protocols rather than improvement of the electrodes themselves. The principal research focus of FES today is improvement of selectivity: the ability of an implant to stimulate a subpopulation of nerve fibers within a nerve. In this way we can activate a function lost to nervous injury or disease, while preventing other fibers from being stimulated and causing parasitic sensations or muscle contractions. Within the constraints of this project, as detailed in section 3.4, it is more appropriate to work on the stimulation method in order to obtain high research impact.

The background provides information on the structure and function of nerves that is essential to understanding the technical challenges of FES. An initial analysis of primary literature and review papers in section 3 highlights the field's major challenges, such as improving electrode selectivity to make fine control of nerves via electrical stimulation a viable treatment option.

In this way the project aims to deliver a new method for nerve stimulation that can be used to improve selectivity in existing electrode technologies. It can be used to treat neurological deficiencies that are as of today not yet or poorly treated. In the example of spinal cord injury, physiological bladder voiding which corresponds to a basic everyday task isn't achievable with current treatment options. The protocol modification is based on the fact that we wish to block large nerve fibers while letting small nerve fibers conduct action potentials. Since action potentials travel along axons at a speed that is approximately proportional to the fiber's diameter, it becomes feasible to block the nerve in such a way that the faster action potentials are annihilated, and the block removed before the arrival of the slower action potentials. In this manner only the slower action potentials traveling in the smaller fibers arrive at their destination, effectively resulting in selective stimulation of small fibers.

For practical and ethical reasons detailed in the background, section 3.4,

computational simulation is the main tool used in this project's research. The project uses the NEURON simulation environment to study the interactions between an axon and two electrodes. The principally studied interaction is that of an external electrode used as a nerve blocking electrode, with the second electrode internal to the nerve as a source of nerve stimulation to provide action potentials for the external electrode to block. COMSOL is used to simulate the electric field generated by the external blocking electrode. However, since the model will eventually need to be validated with experiments on real nerves, an eventual animal model needs to be chosen. The African Clawed Frog *Xenopus Laevis* was chosen due to its availability in the laboratory, the significant amount of past research done on its nervous system, and the relative similarity of its nerves to human nerves. This in turn defined the characteristics of the nerves that the model would simulate, with the Frankenhaeuser Huxley nerve model [13] being used for the channel dynamics within NEURON.

The comparative study of the different nerve stimulation protocols used in past research is relatively recent [22, 7] and there is no computational modeling standard for these studies. Because of this the project also aims to develop a toolbox for researchers to use, that allows efficient and powerful modelization of axons in interaction with extraneural electrodes in nerve blocking studies. A user's manual to this toolbox can be viewed in section 8.1.

In order to clearly define a selective stimulation protocol, the project aims to define the constraints of a hypothetical implant that would allow the use of this protocol, made with existing electrode technologies. The geometrical constraints, and the characteristics of the fibers for which the system is usable are defined in the discussion, section 6. To do this, a preliminary characterization of nerve block was done using the model to define the stimulation parameters at which block was successful (see section 5). With these results, it is possible to predict in which conditions selective stimulation can be achieved. The end product of the project is the selective stimulation protocol, together with the results that make it directly applicable to validation experiments on *Xenopus Laevis*, as well as the software toolbox that can be used to continue research on selective stimulation with more realistic models, particularly for mammalian nerves, and the user manual for ease of use.

# Chapter 3

# background

## 3.1 structure and function of nerves

### 3.1.1 structure of nerves

Nerves are fibrous structures within the body that allow the central nervous system i.e. the brain and spinal cord to communicate with organs such as the stomach, intestines, heart and of course the skeletal muscles. They are organised in bundles of fibers with different diameters (see fig. 3.1, right picture, taken from [6]), branching out to innervate target organs as they are located more distally from the nerve's origin (fig 3.1, left picture, taken from [11]).

Microscopically, nerve cells are composed of a soma or cell body with neurites extending from the cell body to communicate with end organs or other nerve cells. Neurites are dendrites extending a relatively short distance from the soma to receive messages from other nerve cells. One particularly long neurite is the axon and serves to relay the message of the nerve cell to distant locations in the body [6].

In the peripheral nervous system, most nerve fibers, meaning the axons of the corresponding neurons, are myelinated. Myelinated fibers are surrounded by a multitude of layers of myelin produced by surrounding schwann cells (see figure 3.1, right picture). These layers of membrane serve to significantly increase action potential conduction speed. At regular intervals, the myelin sheath is interrupted to expose the membrane of the axon at the nodes of Ranvier [6].

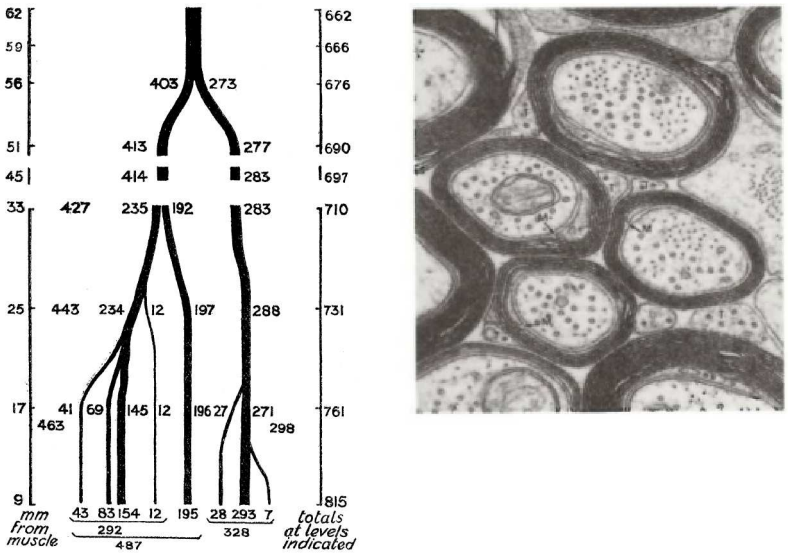


Figure 3.1: Left: Diagram showing progressive branching of the nerve innervating the *gastrocnemius medialis* muscle of the cat, taken from [11]. Right: Myelinated optic nerve fibers cut in cross-section, taken from [6]. Note that the dark portions are myelin and that each axon is myelinated individually.

### 3.1.2 saltatory conduction and the action potential

Nerve cells convey information through their axons by use of electrical signals. In the resting state the axon's membrane is negatively polarised and no current flows occur. When a signal is sent by the nerve cell, the membrane of the axon is depolarised at the soma. Once the depolarisation reaches a threshold, voltage-gated ion channels located in the membrane open and resulting ionic currents make the depolarisation travel along the axon to its extremity, where the message is relayed to end organs or other nerve cells. These propagating depolarisations are called action potentials (see figure 3.2) and rely on ion flows through the axon's membrane to change its potential [6]. The frequency of action potentials conveyed through the axon determines the intensity of the nerve cell's signal.

In myelinated axons, the ion only flows at the nodes of Ranvier where the axon's membrane is exposed, leading to "jumps" of the action potential rather than continuous conduction. This is the effect that speeds up conduction of action potentials in myelinated neurons.

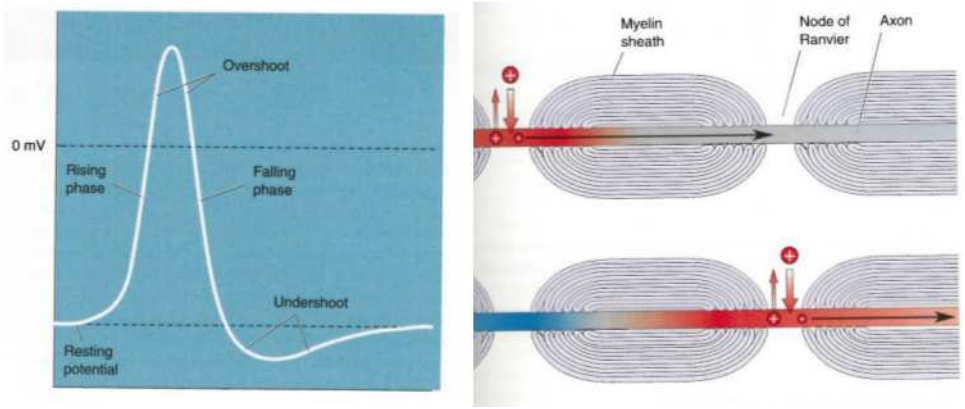


Figure 3.2: Left: Diagram showing the successive phases of an action potential as can be observed from monitoring the membrane potential of a node of Ranvier over time. Right: Illustration showing the action potential jumping from node to node along the axon. Both figures are taken from [6].

## 3.2 spinal cord injury and nerve damage

### 3.2.1 effects of injury

Except for certain autonomic and reflex circuits, all nervous traffic originates from and converges towards the brain [6]. The spinal cord distributes nerves to corresponding end organs from the neck to the tailbone, with nerves exiting the spinal cord at every intervertebral gap (see fig. 3.3). When the spinal cord is damaged, nerves exiting the spinal cord distally from the location of the injury may be severed from the brain and not relay messages anymore. This causes paralysis of skeletal muscle, amongst other undesirable effects [30].

The peripheral nervous system has very poor regenerative capabilities and for this reason once damage has been sustained it is likely to remain for extended or indefinite periods of time, as axons regenerate very slowly (1-3 mm per day). Prolonged denervation of end organs often results in atrophy, compounding the initial damage with complications [36], often making full recovery impossible and motivating research into ways to artifically restore function to the nervous system.

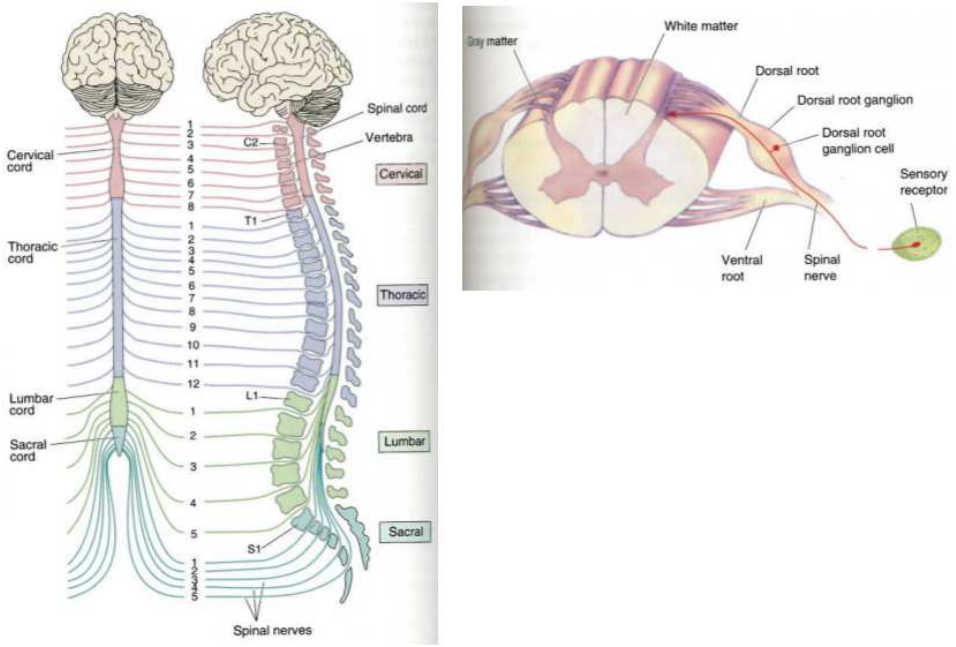


Figure 3.3: Left: Illustration showing the nerves exiting the spinal cord with corresponding locations. Right: Illustration showing the anatomy of the spinal cord. Both figures are taken from [6].

### 3.2.2 case study of bladder control

There are more than 200,000 people in the United States with a spinal cord injury, with approximately 10,000 new cases each year [12]. When spinal cord injury is located in the thoracic or cervical regions, normal bladder function may be lost [32]. In these cases it is common for patients to have a multitude of complications linked to the lost function, such as urinary tract infections, incontinence, and renal failure.

Among the complications, patients often have trouble voiding their bladders. To understand this, we need to look at the muscle system responsible for continence and voiding. Essentially, two muscle groups are of particular interest: the detrusor muscle is responsible for exerting pressure on the bladder in order to void it, and the urethral sphincter is responsible for continence (see fig. 3.4). The muscles are normally activated separately, however during spinal cord injury this coordination is perturbed, a phenomenon called detrusor-sphincter dyssynergia [32]. This prevents normal voiding and leads to sustained high intravesical pressures which can lead to further complications in the urinary tract.

Several proposals to restore lost function have been investigated, amongst which catheterization, which allows the patient to void the bladder at a set frequency by introducing a catheter directly into the bladder [12]. How-

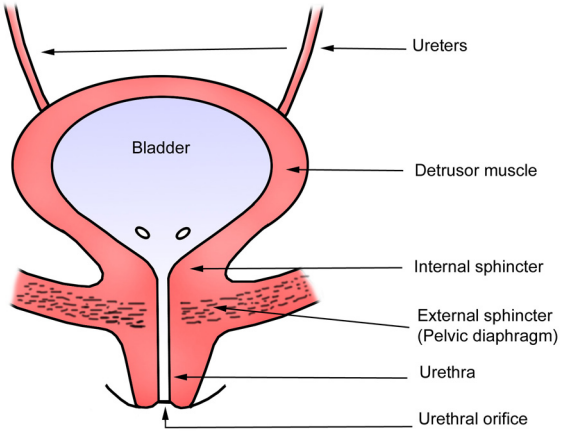


Figure 3.4: Drawing of the gross anatomy of the bladder. Credits to Raymond R Rackley [2], MD.

ever this isn't viable with certain bladder morphologies, or if patient hand function has also been degraded with neural injury. Even with fully automatic systems, chronic catheterization can be traumatic to the urethra or the bladder itself, prompting research into ways to restore bladder function by stimulating the detrusor muscle while avoiding stimulation of the urethral sphincter.

## 3.3 nerve stimulation and block

### 3.3.1 principles of nerve stimulation

It has been known for decades that electrical signals can stimulate nervous activity. Applying cathodic currents to the axon membrane causes its depolarization which then triggers generation of action potentials, just as the membrane would be depolarized when the nerve cell needs to send a signal. With increasing amounts of current, the frequency of generated action potentials increases [6]. Conversely anodic currents hyperpolarize the membrane which can prevent membrane depolarization and thus the passage of action potentials when sufficient current is applied [7].

It has been shown that the origin of stimulation is at the axon [31] rather than the soma of the nerve cell, which is convenient for stimulation of nerves within the peripheral nerve system as the soma is small and often inaccessible when it is located within the spinal cord [6]. As the excitability of nerves with electrical currents is studied, it becomes quickly apparent that this tool can be used to stimulate muscles that were disconnected from the central nervous system by injury or disease, but that still have viable local nerves. A common example is spinal cord injury when the corresponding

limbs haven't been damaged. Application of electrical signals to stimulate nerves and restore function is then called Functional Electrical Stimulation.

### 3.3.2 use of nerve stimulation in therapy

Functional Electrical Stimulation is a well documented tool and its potential for therapeutic application clearly identified from the outset [31]. However, due to the peripheral nervous system's architecture, placing nerves with very different destinations in tightly knit bundles, only a very crude amount of control has been achieved. This is because implants need to selectively stimulate certain fibers within the nerves of the body in order to prevent counterproductive activation of muscles. The corresponding ability of an electrode to stimulate only certain fibers is called selectivity, and it is key to more reliable and efficient treatments in FES [30]. We could think that isolating relevant nerve fibers for implantation would solve the selectivity problem, but the delicate nature of nervous tissue combined with its complex architecture [10] makes the resulting operation impractical if not unfeasible. Hence we must solve the selectivity problem taking into account that unwanted stimulation near the fibers of interest needs to be avoided.

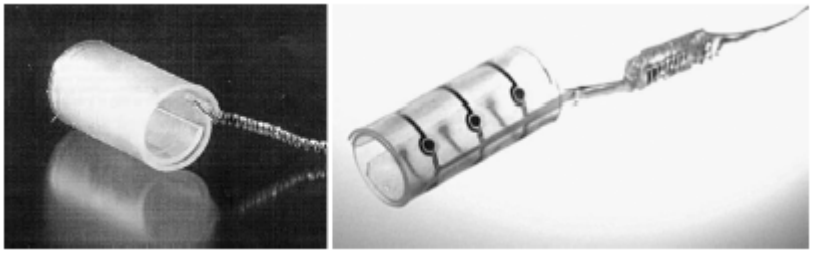

Figure 3.5: Pictures of nerve cuff electrodes where the cylindrical construction that fits over a nerve trunk is clearly visible. Taken from [30].

The drive for selectivity has led to the design of a variety of implantable electrodes [25, 42, 9]. However, save for the "nerve-cuff" electrodes (fig. 3.5) which have been implanted for decades and are the most studied [30], other electrode designs attempt to gain selectivity at the cost of invasiveness [30], which poses several problems in a hypothetical chronic implant. Reliability of FES depends on the electrical properties of the electrode and the tissue in which it has been implanted, and the electrode which has been most studied in chronic implants is the nerve cuff electrode. Results have shown that its input-output electrical properties are stable over the course of these studies [25, 16]. Variation of input-output properties are expected as the body's immune response to electrode implantation leads to the formation of encapsulation tissue, the electrically resistive properties of which have been measured [15]. Since the amount of tissue and its properties depend

on the invasiveness of the implant, it becomes clear that in the scope of this project it is most appropriate to design a method to improve selectivity with nerve-cuff electrodes, which are less invasive than intraneural electrodes.

In the case of bladder control after spinal cord injury, the most promising site for stimulation to restore function is where the sacral roots leave the spinal cord [32], but unfortunately the nerve there contains fibers that lead both to the detrusor muscle and the urethral sphincter. The fibers differ in size with the larger ones leading to the urethral sphincter. Unfortunately, one of the fundamental problems of FES is the inverse recruitment of nerve fibers: whereas during normal function nerve fibers are activated by the nervous system from small to large diameters as the stimulus is graded in intensity, with FES the order is inversed and large fibers are always stimulated first causing increased muscle fatigue and decreased function [32, 35, 8]. As a consequence, when the nerve is stimulated with conventional electrodes and protocols, both the detrusor and the urethral sphincter muscles are stimulated, leading to suboptimal voiding in spurts.

This is a classic example of where selective stimulation based on fiber size would be useful, and so this project aims to find a candidate solution to improve treatment in these cases.

Amongst the possible solutions proposed in Rijkhoff et al's work [32], we will focus on the ones involving selective activation of nerve fibers smaller than the ones in their vicinity, theoretically resulting in contraction of bladder muscles while avoiding closure of the urethral sphincter. All of these techniques involve some form of block to prevent the larger fibers from being stimulated.

### 3.3.3 nerve block

Selectively blocking nerve fibers seems the most appropriate approach to the problem in the context of this project as it is directly linked to stimulation protocol. While there exist chemical or mechanical block of nerves using drugs or pressure to prevent conduction, these aren't applicable to our issue since they are either too slow to be practical for on-demand use or incur damage to the nerve when used chronically. However electrical block has shown great promise in that it is fast acting and reversing, and that it can be used chronically under certain conditions. There are two principal types of electrical block: AC block [22], and DC block [7], both reviewed by Kilgore and Bhadra in 2004.

AC block is established and removed within a few milliseconds of blocking signal activation and cutoff, under the right conditions. The identified mechanism is a widely spread depolarization of the membrane in the steady-state that prevents action potential conduction[22]. However, AC block produces a characteristic "onset reponse", corresponding to a period of intense neuronal firing after the blocking signal is started. This period

is of variable length depending on experimental conditions with no clear explanation of the underlying mechanisms, with two phases. Phase 1 is of very short duration and corresponds to a single muscle twitch. It cannot be avoided with conventional blocking techniques, however several papers have reported methods to reduce or nullify onset reponse [14, 2]. Phase 2 can last tens of milliseconds, and it the most variable phase depending on experimental conditions. Most importantly, researchers have successfully used AC techniques to selectively stimulate nerve fibers based on their size [35, 21, 1]. These techniques rely on the fact that, similarly to how large fibers are more readily stimulated than small nerve fibers at a certain stimulus amplitude, they are also blocked first, allowing smaller fibers to keep conducting.

DC block is also fast-acting and fast-reversing, to the order of the tenth of a second. Like AC block it also blocks large axons first, making selective stimulation a possibility [7]. The principal identified mechanism is inactivation of sodium channels necessary for action potential propagation by sustained membrane depolarisation. While selective block can be achieved by carefully tuning the blocking signal amplitude, the most interesting blocking technique cited by Bhadra and Kilgore is the one used by Pflüger in 1858 which consisted in a timing block, using the difference in action potential propagation speeds between small and large fibers to block the faster-conducting action potentials. This is particularly interesting because as a general rule action potential propagation speed increases linearly with fiber diameter [10], allowing for use of the technique while the whole nerve is blocked: by stimulating all fibers proximally, action potentials can be filtered by propagation speed, with a fiber size cutoff. This technique was used again by Kuffler to study small nerves which could not necessarily be isolated from their larger counterparts [24], and he includes a diagram of the preparation that explains the principle of the technique (fig. 3.6).

However, there are several major drawbacks to the technique and with DC block in general. In DC block the nerve is stimulated when it is subjected to current steps, a phenomenon called "make" and "break" excitation. Even though this can be remedied with custom DC blocking waveforms [7], there remains the essential problem of DC block in that it has been shown to damage nerves and is thus unsuitable for any kind of chronic application [43]. This has been attributed to electrochemical reactions occuring at the electrode and the blocked nerve that results in damage of both during extended use of DC. Because of this a set of rules for safe electrical stimulation of nerves has been made [27] and needs to be taken into account. The rule set states that it is always preferable to use the smallest stimulus amplitude in order to achieve the desired effect, making DC unsuitable for use with current electrode technologies, even though work has been done to create an electrode which avoids DC-induced nerve damage [3]. The conclusion is then that AC blocking techniques must be used over DC based techniques. Since AC block has been reported to require higher blocking signal amplitudes at

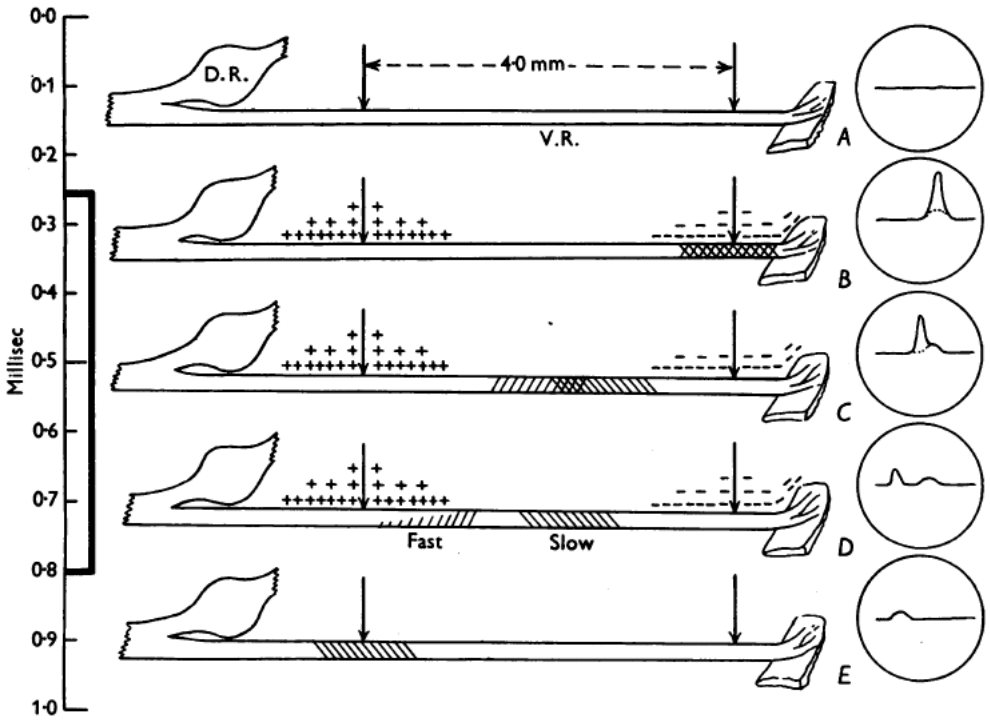


Figure 3.6: A drawing of the nerve preparation used by Kuffler with a DC timing block. The figure explains that the faster action potentials are annihilated at the anode before the block has dissipated, while the slower action potentials are allowed to conduct. Taken from [24].

higher frequencies [22], it is then necessary to characterize the block in order to find the lowest frequency at which it can reliably occur.

## 3.4 problem approach

### 3.4.1 blocking technique modification

The most promising technique for selective stimulation is adapting the DC timing block to use AC in a safe and reliable way, for several reasons. First of all, using AC will prevent any DC-induced nerve damage by enforcing charge-balanced blocking waveforms, making the technique viable for chronic use and with existing, FDA-approved electrode technologies. As nerve damage has been reported to reduce action potential conduction speed [43], preventing this damage makes a reliable timing block feasible. Then, even though it is possible to selectively block larger fibers by adjusting the blocking amplitude, this is complicated by the progressive formation of scar tissue, the extent of which can depend largely on the implanted individual.

This tissue affects electrode input-output properties [16, 15] and can thus change the thresholds at which block is achieved, generally making them higher. This source of unreliability can be avoided altogether with the timing block technique provided that the nerves aren't damaged, since that ensures the stability of action potential conduction speed. In turn this gives us fine control over the maximum fiber diameter which we want to allow to conduct, making graded recruitment of fibers from small sizes to large sizes theoretically possible.

### 3.4.2 choice of an animal model: *Xenopus Laevis*

As it would be unrealistic to attempt clinical experimentation on humans to research a better blocking protocol for selective stimulation during this project, the two choices left are either animal experiments or modelization. Nerve block research using animals generally excises a nerve of interest from the animal, attached or not to a muscle depending on the method for nerve output measurement. This is done under anesthesia and the animals are routinely sacrificed after the procedure [23, 21, 24]. Due to ethical and financial concerns for the project, as well as the unknown amount of experiments needed to design a reliable blocking protocol for selective stimulation, it seems most appropriate to investigate the issue using modeling as the main tool, with eventual use of animals for validation of results. Since it will be of interest to verify the project's findings with animal experimentation at a later date, an appropriate animal model has to be chosen. The laboratory at Imperial uses the African Clawed Frog *Xenopus Laevis* for animal studies. It is one of the major animal models in neuroscience, meaning that its neurophysiology is very well documented, featuring nerves with relatively longer lifetimes once excised [41]. The Frankenhaeuser and Huxley (FH) axon channel dynamics model was made from patch clamp recordings of *Xenopus Laevis* neurons [13], giving us the necessary equations for modelization of myelinated nerve, similar to that found in the human peripheral nervous system. Furthermore, the fibers found in its ventral roots can be separated into two categories: one for large fibers of about 12 microns in diameter and one for small fibers of about 5 microns in diameter [24], in a configuration similar to that of a human's sacral roots. Since the setting is similar, selective stimulation of the small fibers in the frog's ventral roots could serve as a proof of concept for an improvement of sacral root stimulation. Because of these reasons, *Xenopus Laevis* was chosen as the animal model for the project.

### 3.4.3 choice of a nerve model: Frankenhaeuser-Huxley

The FH model is a model for the ion channel behaviour of frog neurons within an axon cable model. In cable models, the axon is treated as an

electrical cable with the membrane acting as a capacitor and its cytoplasm, here named the axoplasm, treated as a resistor. The membrane's capacitive nature allows it to be polarised, which is an essential property of axon membranes and necessary to explain action potential conduction. Voltage-gated ion channels are resistors in parallel with the membrane's capacitors, that allow current to move between the inside and outside of the cell. The behaviour of these channels changes their resistance as a function of the membrane's polarisation, that is the potential difference between the inside and outside of the cell. A description of the model using an electrical diagram can be viewed on fig. 3.7.

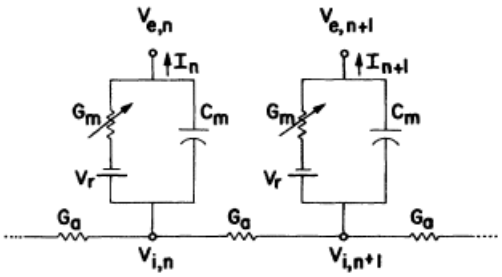


Figure 3.7: An electrical diagram corresponding to the FH model, that describes the lumped circuit analogy of myelinated axon. Myelin is treated here a perfectly insulating, hence the lack of any transmembrane leakage resistances between nodes. Taken from [26].

One of the first channel models was that of Hodgkin and Huxley 1952, made from measurements conducted on squid axons. However the FH model is better suited for modelising frog nerves, since it is based on measurements of the nerves of the African Clawed Frog *Xenopus Laevis* [13], which is the model animal we chose for the project. The FH model features 4 types of channels: a fast sodium channel, a persistent sodium channel , a potassium channel, and a non-specific channel, with different contributions to the transmembrane current during action potential conduction. It is these channels that are responsible for the excitability of the nerve membrane, just as they are responsible for the blocking mechanisms that can prevent action potential conduction.

### 3.4.4 simulation software tools: NEURON and COMSOL

In order to carry out the simulations, an appropriate tool must be chosen. To this end the NEURON simulation environment is specially tailored to these types of studies with the powerful HOC and NMODL [19] programming languages. The problem approach is modular and the physiological part of the nerve model, such as the ion channel dynamics, is separate from the

model set up, geometry, and experimental protocol. This has the advantage of easy switching between human or mammalian nerve models and frog nerve models, for example. In this way a robust selective stimulation protocol can be easily adapted and tested on different species. NEURON also lends itself well to the creation of a software toolbox for researchers to use in the context of selective stimulation, which is another one of the goals of this project as there aren't any standard model implementations for selective stimulation using timing blocks. To make the electrode geometry independent from the model itself, the Finite Element Modelisation program COMSOL can be used to compute the electric fields produced by a certain electrode, and these fields output as files that are readable by NEURON. Fortunately, the FH model is already implemented [17] [Hines, Senselab] in NMODL and so can be directly used in conjunction with HOC files describing the model geometry, setup and the stimulation protocol.

# Chapter 4

# methodology

## 4.1 model geometry

Now that we have specified the context of the model we must choose an appropriate model geometry, and a first draft of the stimulation protocol that results in selective stimulation. The timing block mentioned in the literature can be modelised with an axon featuring a stimulation electrode at its proximal end, and a blocking electrode somewhere along the axon, placed in such a way that it is possible to check for the presence of an action potential distally from the blocking site. The stimulation electrode would provide a source of action potentials that propagate distally towards the blocking electrode. The blocking electrode would be used to time a block that would result in the action potential being annihilated or passing through, depending on its propagation speed.

To simulate cuff electrodes around a nerve containing multiple fibers of different sizes, two axons are modeled, each with a different diameter. In this way a "small" axon can be identified relative to a "large" axon. The locations of the stimulation and blocking electrodes would be the same on both axons, which we model as having the same length, similarly to how a nerve would be prepared in an animal experiment. In this way the model geometry is directly related to a hypothetical animal experiment that could be done to verify simulation results. A schematic of the model geometry can be found below, fig. 4.1.

It can be observed on fig 4.1 that the two modeled fibers have different diameters while having the same length of 3 centimeters. This length was chosen as a typical length for nerve preparations ([24] uses preparations of at least 40 mm in the frog), and also to give some leeway in the placement of the blocking electrode should extra distance between the stimulation and blocking electrodes be needed.

The simulation will rely on the difference in action potential conduction speeds to perform a fiber size-selective AC block at the center of both axons.

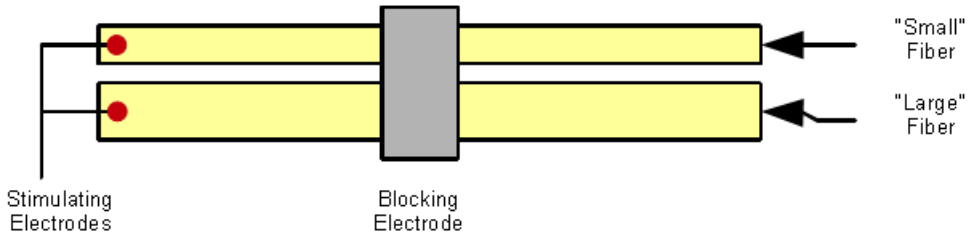


Figure 4.1: A stylised drawing showing the model geometry with a "small" and "large" axon. Intracellular electrodes at the left end are responsible for action potential generation, and the cuff at the center is responsible for blocking.

The protocol assumes that the blocking electrode blocks all fibers in the nerve so that there isn't any parasitic firing that could interfere with the selective stimulation. With a set time at which the blocking signal is cut off, a stimulation pulse is then timed so that the faster-propagating action potential is annihilated at the block site, while the slower-propagating action potential is successfully conducted to the distal end of the small axon.

## 4.2 first iteration of the model

The first iteration of the model was purely to test if the timing block with AC stimulation was possible. The model was kept extremely simple with an unmyelinated axon, and intracellular electrodes. This means that the effects of myelin, which speeds up conduction and "discretizes" the position of action potentials, were absent from this model. Intracellular electrodes directly inject current into the cell body, and so there aren't any extracellular electric fields to modelize yet with COMSOL, and the current source is a point at the center of the axon, limiting the spread of the blocking current. It is evident that the final iteration of the model will have to address all the limitations of this first model, but the hypothesis that timing block using AC was possible to obtain selective stimulation needed to be tested first in an idealized situation.

The simulation used two axons of 5 and 12 microns in diameter respectively with intracellular electrodes located at the left (proximal) end and middle. The axon diameters were obtained from [23] and [24], corresponding to the diameters of frog nerve fibers of the ventral roots. The selective stimulation protocol was as follows:

1. Turn on the blocking signal by imposing a sinusod of 3000 Hz at the center electrode, of peak to peak amplitude 400 nA.

2. Wait 5 ms to allow the block to stabilize

3. A t = 12.5 ms, send 0.12 ms pulse through the left electrode to generate an axon potential in both axons[1].

4. Time the block cutoff at 19 ms to annihilate the faster action potential.

5. Verify that the action potential propagating in the larger fiber has been annihilated, while the action potential of the smaller fiber has conducted through the block once it has been cut off. If these conditions are true then selective stimulation has been achieved.

The block signal frequency, amplitude and cutoff times were all determined empirically to produce selective block, since this was an idealized situation where the theoretical possibility of selective stimulation was tested. The results of the simulation can be seen in subsection 5.1. As the results indicate that selective stimulation of the smaller fibers can be achieved in this simplified model, a more realistic model with myelinated axons and extracellular fields generated by the blocking electrode can be used. As the timing parameters for the stimulation and block cutoff were determined entirely empirically, the new model will also use more properties of the axons of *Xenopus Laevis* in order to automatically determine correct stimulation and block cutoff timings to obtain selective stimulation. It needs to be noted however, that at this point in time no characterization of the block itself had been made, and that it was possible that the smaller fiber had not in fact been blocked and that the selective block was of the threshold type, rather than the timing type. This was eventually remedied with the more realistic model below, and a more robust protocol.

## 4.3 second iteration of the model

The second model keeps the original geometry with the two modeled axons of 30000 microns in length, with the 5 micron and 12 micron fiber diameters by default. However, batch runs where parameters such as fiber diameter can be changed from one simulation to the next are possible, since we can suppose that there are fibers of diameters anywhere within the 4 to 15 micron range[23, 40], and that we must test the selective stimulation protocol to determine if it is possible to differentiate fibers that are closer in size to each other. For this model, the physical and physiological aspects of the simulation were separated as follows:

- The blocking electrode is now modelized as a monopolar electrode surrounding the axons, of 1 mm in width and 1 cm radius. The physical

[1]The amplitude was 100 nA for the 5 micron fiber and 240 nA for the 12 micron fiber, taking into account the larger cross-sectional area in order to get the same stimulus amplitude in terms of membrane potential variation.

model for the electrode was implemented with cylindrical symmetry in COMSOL in order to obtain the distribution of the component of the electric field that was perpendicular to the fiber, over the fiber itself. From an output file that maps the electric field values as a 2D matrix, the values at a certain radial distance from the electrode are used within NEURON, with a default electrode to fiber distance of 1 mm, as in [22].

- The fiber is now myelinated with myelinated sections treated as perfectly insulated compartments: current inside can only flow within the axoplasm. The properties of the myelinated sections were obtained from [28, 38], and the myelin sheath is considered perfectly insulating with regards to the electric field. The ratio of internode length in microns to fiber diameter in microns is constant and the value is 100.
- The nodes of Ranvier closely follow the potential specified by the external electric field, which is considered directly linked to the membrane.
- The selective stimulation protocol ensures a charge-balanced waveform and prevents stimulation of the fiber when the blocking signal is cut off[2].

As this second model represents the selective stimulation problem much more realistically, it must be characterized and validated. As direct experimental validation on frogs was not feasible during the project, validation using existing experimental data from the literature is the next most reliable method. Validation consisted in comparing the model's predictions to experimental data in the case of blocking thresholds for frequency and amplitude, as well as action potential conduction speeds, since these are the two most important neurophysiological parameters in timing blocks, as explained below. If the model is validated in this way it is safe to assume that it correctly simulates the behaviour of nerve fibers with regards to selective stimulation using AC timing blocks. This serves a dual purpose, as once the model's simulation of block is characterized, the data can be used to select optimal parameters for selective stimulation, for maximum reliability and safety.

## 4.4 model characterization

For reliability we must suppose that we want to avoid any firing of nerve fibers that isn't expressly stimulated. This implies using the maximum am-

[2] A phenomenon akin to anodal break stimulation [33] occurs when the blocking signal is cut off in the middle of the depolarising part of the oscillation: one of the software package demonstrations showcases this effect. It can be corrected, while maintaining exact timings on blocking signal cutoff by adjusting the stimulation timing instead (see User's Manual, section 8.1).

plitude for block at a frequency which results in electrical block in all fiber diameters of the implanted nerve. Then there must be the longest distance possible between the stimulation and blocking electrodes of the implant, so that even small differences in conduction speed lead to large differences in the times of arrival (TOA) of the action potentials of fibers with differing diameters, giving the widest time window during which the block can be cutoff to allow passage of the slow action potential, while annihilating the faster action potential. However safety procedures suggest that the lowest amount of charge be injected per cycle in the blocking signal [27, 22] and that the implant be the least invasive, and so the smallest possible. It then becomes apparent that a tradeoff between these aspects must be considered in the design of an implant using the timing block technique, and that precise characterization of the behaviour of blocked fibers is needed: we need to know the blocking signal amplitude and frequency thresholds at which the fibers can be reliably blocked. Furthermore, to time a stimulatory pulse and obtain selective stimulation we must know the action potential conduction speed. The results of characterization can be viewed in section 5, subsection 5.2 [3].

### 4.4.1 action potential conduction speed

The first aspect of the model that was characterized was the conduction speed, as it was easier to compare the simulation data to existing experimental data and from there to make corrections to the model if necessary. The protocol that was used is as follows:

1. No block is applied to the nerve fiber and action potentials conduct freely accross the axon.
2. A single action potential is sent from the left end of the axon, propagating towards the right end.
3. The times at which the action potential is detected at the left end and the middle of the axon are recorded by NEURON [4].
4. The program calculates the distance between the two aforementioned points to give the resulting propagation speed.

[3] The results were obtained with a version of the model only representing one fiber since this is all that is needed when characterizing the model.

[4] This is done by APCounter objects placed at the corresponding points on the axon. These virtual detectors record the times at which the membrane potential at their location crosses a threshold from a lower value to a higher value than the threshold. In this way action potentials are detected only once during their passage.

### 4.4.2 block threshold

For block threshold characterization, a few details about how the electric field used for simulation of extracellular electrodes was obtained are needed. The blocking electrode was modelized as a ring of 1 mm width and radius 1 cm within COMSOL. The large radius was primarily chosen to make sure the field was not perturbed by boundary conditions. The extracellular medium was modeled as purely resistive and anisotropic with a radial resistivity $\rho_r = 560\Omega$.cm and a longitudinal resistivity $\rho l = 356\Omega$.cm as specified in [4], in an analysis of Tasaki's measurements with frog nerve trunks [39]. Even though the modeled axons are 3 cm long, the length of the cuff within COMSOL is 10 cm because this allows the electrode to be modeled at different longitudinal positions over the length of the axons, as well as changing the axon length (max. 5 cm if all electrode positions are to remain possible) within the model. The output current of the electrode was set at 159 A/m$^2$ in order to obtain 1 mA for a 1 mm radius cuff. Since the cuff has 1 cm radius within the model, the actual modeled output current is 10 mA. The electric field generated by the electrode(s) is output by COMSOL as a file describing a surface, and a graphical representation of the electric field distribution for both monopolar and bipolar[5] electrode configurations can be found section 5.2, figures 5.4 and 5.5. NEURON then extracts the values for a certain distance from the electrode and applies them as an amplitude coefficient to the sine wave for the blocking signal. As the extracellular medium is considered to be linear and was verified to be so within COMSOL, the electric field values are modeled for a sine signal of amplitude 1 mA. We can then adjust the amplitude of the blocking signal within NEURON by applying a dimensionless coefficient. The equation for the amplitude of the sine wave for the extracellular electric field at a point on the modeled axon within NEURON is thus:

$$A = E_{Value} * C$$

With A as the amplitude in volts per meter, $E_{Value}$ as the electric field value at the point considered along the axon in volts per meter and C the amplitude coefficient corresponding to the amplitude of the electrical signal at the electrode in milliamps (an approximation: see above for the influence of model geometry). This amplitude is modulated over time by a sine wave of specified phase and frequency. This more realistic model of nerve block also correctly simulates onset response with variable numbers of action potentials being generated before block is effectively achieved, depending on blocking signal parameters. The protocols thus have to take this onset response time into account by ignoring action potentials generated during this

[5]At this point in time only monopolar electrode configurations were used in the simulations for practical reasons explained below in this section, and bipolar configurations were then used for comparison afterwards.

time with regards to identifying if a block is successful or not. Unless stated otherwise, all simulations were done with an electrode-fiber distance of 1000 microns, which corresponds to the furthest distance a fiber can be from the electrode in a hypothetical 1 mm radius cuff, able to fit over the nerves of *Xenopus Laevis*. Characterization of blocking thresholds was carried out with the following protocol:

1. The blocking signal specified by user parameters is applied to the axon at its center.
2. The simulation waits for the block to stabilize by a set amount of time specified by the user as the onset response time (20 ms are used by default).
3. A train of five action potentials are generated with intracellular current impulses of 10 nA for 0.15 ms delivered to the left end of the axon, separated by 5 ms intervals.
4. The left and right ends of the axon are monitored for the passage of action potentials.
5. Any action potentials detected before the onset time has passed are ignored.
6. If the action potential count is 5 at the left end and 0 at the right end[6] of the axon, the block is considered successful, else it is considered a failure.

## 4.5 selective stimulation: timing of the stimulus

After the properties of single axons have been characterized, additional data is needed to ensure that selective stimulation of small fibers can happen with parameters automatically determined by the protocol, that is by a hypothetical implant using the timing block technique. The success of selective stimulation rests on the ability of the device to block all relevant fibers in the implanted nerve, and to precisely time a stimulatory pulse so that when the blocking signal is cut off, only slow action potentials propagating in small fibers are allowed to conduct. Considering this, we need a method to calculate this stimulation timing.

Since the protocol requires that the block be stabilized before cutoff and stimulation, the time at which the block can be cut off is known and noted as $t_{bco}$. However there is a time after the blocking signal is turned off

[6] 5 APs were generated and all must be accounted for proximally, and none of them must be detected distally.

during which any action potentials arriving at the location are annihilated. This time is analogous to the axon's refractory period, but since it concerns blocking it will be named block dissipation time and noted as $t_{bd}$. From the sum of these two times, the time necessary for the slow and fast action potentials to arrive from their origin point can be substracted, giving the time at which the simulatory pulse must be sent. These times are noted as $t_{ps}$ for the propagation time of the slow action potential and $t_{pf}$ for the propagation time of the fast action potential. Since these times are different, we obtain a window during which we must send the stimulation pulse in order to obtain selective stimulation. A diagram showing the timing window for selective stimulation can be seen fig. 4.2.

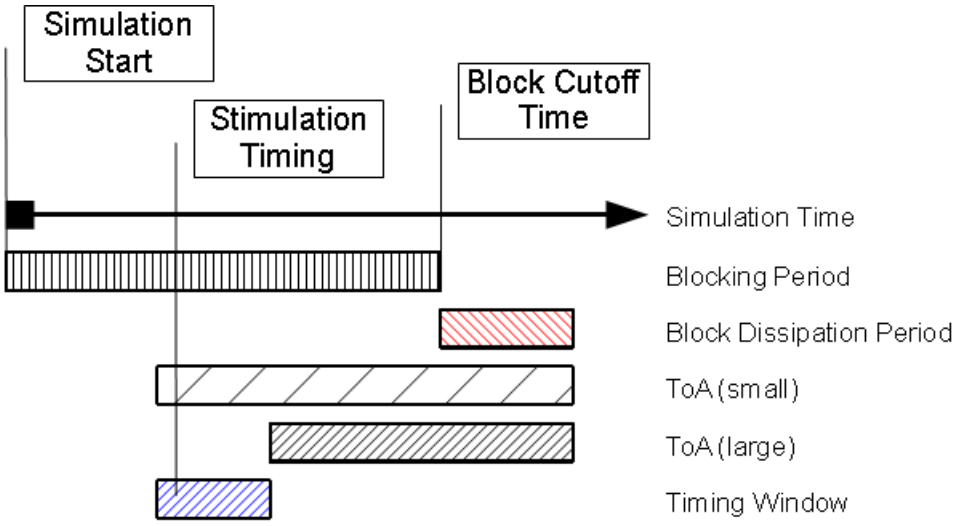


Figure 4.2: A diagram showing the different times taken into account when calculating the time at which the stimulation pulse needs to be sent. The algorithm for timing chooses the middle of the timing window.

With the algorithm choosing the middle of the timing window, it ensures that the fast action potential arrives at the blocking area before the block has dissipated, ensuring its annihilation, and in the same vein ensures the passage of the slower action potential after the blocking effect has dissipated. The equation for the timing of the stimulatory pulse is thus:

$$t_{pulse} = t_{bco} + t_{bd} - \frac{t_{ps}+t_{pf}}{2}.$$

As the block dissipation time $t_{bd}$ was not known explicitly, though it was likely to be of the same power of ten as the refractory period of nerves which is of about 2 ms in the frog [40], the block dissipation time needed to be measured. This was even more important as according to Tasaki's paper, it could depend on fiber size. The dissipation period was determined

by placing the blocked area proximally, so that stimulated APs had to cross the area just after block cutoff. In this way the influence of propagation time was nullified, and the dissipation time was the minimum time between block cutoff and stimulation time for there to be a propagating action potential reaching the distal end of the fiber. The measurements can be found in section 5.2, table 5.2.4.

From these measurements, a protocol for selective stimulation was created:

1. At the start of the simulation, the same extracellular blocking signal is applied to both axons.

2. The simulation waits for the block to stabilize by a set amount of time specified by the user as the onset response time (20 ms are used by default).

3. A single action potential is generated with a current impulse of 10 mA for 0.15 ms delivered to the left end of the axon, at a timing determined by the aforementioned calculation.

4. The blocking signal is cut off after an even number of oscillations to ensure charge balance.

5. The right end of both axons is monitored for the passage of action potentials.

6. If the action potential passed the block in the small axon, was annihilated in the large axon, and that there weren't any unexpected action potentials generated during the simulation except for the onset response action potentials, then selective stimulation is considered successful. The program uses recorded action potential propagation speeds to calculate the TOA for the generated action potential in the small axon, and compares it to the recorded time; in this way even if an action potential is generated by the block, it cannot fool the program into giving a false positive.

It is this protocol that is used in all the selective stimulation simulations done hereafter.

## 4.6 constraint determination: interelectrode distance

As the block had been characterized, what remained to be determined was the other constraints for selective stimulation to be successful. These essentially depended on the timing, which was determined by the diameter

difference between the small and large fibers, the distance between the stimulatory and blocking electrode (interelectrode distance), and the electrode-fiber distance, since all the previous tests had been done with a constant electrode-fiber distance of 1mm, and that depending on the diameter of an implanted nerve, this distance could vary. Together, the tests for these constraints define the geometry of a hypothetical implant using a timed block for selective stimulation.

One of the most important aspects of the technique is its ability to filter out action potentials propagating in larger fibers. An evident question then is how much larger does the larger fiber have to be in order for the technique to work?

Provided that electrical block is achieved, since the timing procedure depends on the action potential propagation times $t_{ps}$ and $t_{pf}$, and that these depend on fiber diameter and interelectrode distance, supposing that both $t_{bco}$ and $t_{bd}$ remain constant, we can measure the protocol's resolution, defined here as the minimum fiber diameter difference at which selective stimulation is reliable. To measure this, selective stimulation tests were done with interelectrode distance and fiber diameter difference as the variables. The large fiber's diameter was fixed at 12 microns and the small fiber's diameter varied from 5 to 12 microns in steps of 0.5 microns. The interelectrode distance varied from 0.75 cm to 4.25 cm in steps of 0.25 cm, with a final uncertainty on the results equal to the step for the two variables respectively. The case where both fibers had the same diameter served as a negative control as it is then theoretically impossible to selectively stimulate one of the two fibers if the model correctly simulates the same situation in both fibers. As modifying the large fiber's diameter instead of the small one's would lead to larger relative differences in diameter at small diameter differences (e.g. 5 microns and 6 microns instead of 11 microns and 12 microns), the experiment simulating the smallest relative differences was chosen by default[7]. Amplitude and frequency for the blocking signal were chosen based on the graph for 5 micron fiber blocking thresholds, at 5 mA and 3300 Hz respectively. The test supposed that if the 5 micron fiber was blocked, so would larger fibers, based on the aspect of block threshold plots for fibers with a diameter larger than 5 microns, which all have broader intervals of parameters resulting in successful blocks. The results of the simulation can be found in section 5.3, fig. 5.12.

[7]Since the conduction speed curve shows a straight line, the conduction speed difference between two fibers of differing diameters should only depend on the diameter difference.

## 4.7 constraint determination: maximum implantable nerve diameter

In the same vein, since the study of the influence of interelectrode distance on the reliability of the technique defined the minimum cuff length for the implant, constraints on the cuff diameter also had to be defined. These depend on the influence of electrode-fiber distance. Since the technique relies on a total nerve block in order to avoid any parasitic firing of large fibers and complete control of the firing rate of the small fibers, we must consider the worst case scenario, that is to say the fibers lying at the center of the implanted nerve. Since small fibers are inherently more difficult to block than large fibers, we must also consider the smallest fiber diameter for which the technique is applicable. Due to the results obtained for block characterization of 4-micron fibers (see section 5.2, fig. 5.6 and chapter 6), a 5-micron fiber was simulated in this test.

When increasing the electrode-fiber distance, it is the block efficiency which is the most important measure to consider. As explained in chapters 5 and 6, when neither fiber diameter difference nor interelectrode distance change, if block of the smaller fiber is possible, block of the larger fiber is also possible and from there, selective stimulation of the smaller fiber is possible. Hence the test is a frequency and amplitude block threshold test for several different electrode-fiber distances, because the result is information on whether the block of the smaller fiber is possible.

As the electric fields generated by a monopolar electrode on the fiber are the only changing parameter[8], these were plotted for electrode-fiber distances of 750, 1000 and 1250 microns on fig. 5.13 (subfigures A, B and C respectively), section 5.3. To correlate the electric field distribution data with block efficiency, block threshold simulations were then carried out for electrode-fiber distances of 750 and 1250 microns since data had already been obtained for 1000 microns. The protocol used was the same as previously, and the results can be found section 5.3, figures 5.14 and 5.18.

Comparison of curves indicates that for smaller electrode-fiber distances, block is easier to achieve and so there wasn't any further investigation of block thresholds for distances smaller than 1000 microns. However for distances superior to 1250 microns block becomes difficult or even impossible for 5-micron fibers, prompting additional simulations for electrode-fiber distances in between 1000 and 1250 microns, shown figures 5.15, 5.16 and 5.17 for 1050, 1100 and 1150 microns of electrode-fiber distance respectively.

Geometrical constraints imposed on a hypothetical system using the timing block technique, according to previous simulations, are that the implanted nerve cannot have a radius longer than 1100 microns. This is primar-

[8] Amplitude and frequency change within a batch simulation, but all simulations test the same combinations of frequency and amplitude.

ily due to blocking thresholds of small fibers as large fibers with diameters superior to 9 microns remain blockable at 1100 microns of electrode-fiber distance (see fig. 8.3 in section 8.2).

## 4.8 tentative experiments with a bipolar blocking electrode

However, it should be noted that so far results have been obtained with the electric field distribution from a monopolar electrode, since it was then much more straightforward to define the position of the electrode over the axon, an essential parameter for selective stimulation. In a more realistic scenario, electrodes within a cuff are bipolar or tripolar, and since it has been reported that bipolar block is more successful than monopolar block[22], a series of simulations for block thresholds using the same aforementioned protocol[9] were carried out, but for a bipolar electric field distribution.

Bipolar electrodes were implemented in COMSOL with two 1 mm width electrodes identical to the monopolar electrode, spaced 1 mm appart. The resulting changes on block thresholds can be seen figure 5.13, subfigure D for the electric field distribution generated by a bipolar electrode configuration, and figures 5.19, 5.20, 5.21 and 5.22 for block thresholds for 4, 5, 6, 9 and 12 micron fibers respectively in this configuration[10]. Due to the presence of two electrodes in the bipolar configuration, it is more difficult to define the location of the blocked area and from there, to precisely time a stimulus in order to obtain selective stimulation, and it can be expected that the selective stimulation protocol will be ineffective with bipolar electrode configurations. A simulation to ascertain this was carried out with results shown fig. 5.23. The success test conditions had to be relaxed as it was observed that the larger blocked area significantly reduced the conduction speed of incoming action potentials, resulting in false negatives due to the timing test in the model to ascertain if the recorded action potential was actually the one that had been originally stimulated (see section 8.1). The refractory period of the nerve also had to be lengthened to 1.6 ms as opposed to the previous value of 1.4 ms for monopolar block in order for the stimulatory pulse to be timed correctly.

The collated results, once analysed, yielded a proposition for a robust selective stimulation protocol in the conclusion, section 7.

[9] It should be noted here that the protocol is the same but that the phase of the blocking signal had to be changed to $+\frac{\pi}{2}$ to avoid break stimulation and make detection of successful blocks more straightforward.

[10] As 15-micron fibers had already been tested with a monopolar electrode and the block threshold trends with rising fiber diameter already established, less graphs are necessary to establish the same trends for bipolar blocking electrode configurations.

# Chapter 5

# results

## 5.1 initial

This subsection deals with the initial results obtained with the simplified model. The goal of the simulation was to know if selective stimulation based on an AC timing block was possible in an idealized situation. [1]

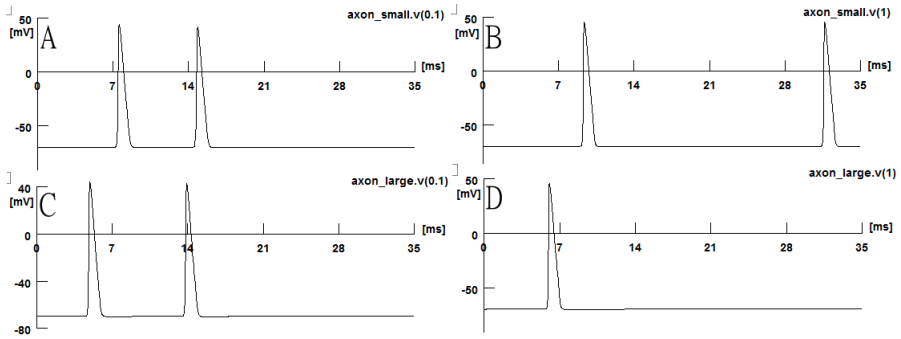


Figure 5.1: A series of time-courses showing the membrane potential as a function of time at different points on the modeled axons. In NEURON the positions marked for monitoring are relative to the axon's length, hence here v(0.1) means the membrane voltage near the proximal (left) end of the axon and v(1) means the membrane voltage at the distal (right) end of the axon. A and B represent membrane voltage in the small axon, C and D membrane voltage in the large axon.

In the simulation shown fig. 5.1 the blocking signal at the center of the axon was turned on at t=0 ms and maintained until t=19 ms. Two action

[1]The situation here is idealized because the block is extremely localized, the currents injected are precisely controlled, and the lack of myelination slows action potential propagation, making it easier to time the block. None of the aforementioned properties are true when switching to a more realistic myelinated axon model.

potentials propagating in different directions were generated at the block location and are detected first by the virtual sensors at the axons' extremities. This is the onset response. The simulation waits until t=12.5 ms to avoid the effects of the refractory period and then generates an action potential at the left end of both axons. This action potential is quickly detected by the leftmost detectors, with the large axon's detector registering the action potential first since the difference in diameters causes the aciton potential to propagate faster in the larger axon. At the distal end of both axons, only the detector on the small axon registers the passage of the generated axon potential, meaning that the faster action potential in the large axon has been annihilated by the block, while the slower action potential in the smaller axon was let through. Selective stimulation of the small axon has thus been achieved, and the project is able to move on to a more realistic model.

## 5.2 characterization and validation

### 5.2.1 conduction speed

The first step in the characterization of the model was verifying if the conduction speed of simulated action potentials agreed with the literature. A test program was made to record the conduction speed of fibers from 2 to 20 microns in diameter, as the interval corresponds approximately to the minimum and maximum reported diameters of myelinated fibers that can be found [34, 40, 20]. The simulation curve is directly compared to measurements of nerve conduction speed in *Xenopus Laevis* by Hutchinson et al [20] fig. 5.3, while the original simulation curve can be viewed fig. 5.2.

From the figures it can be concluded that there is a good quality fit of the model's predictions with the experimental data. See section 6 for more details.

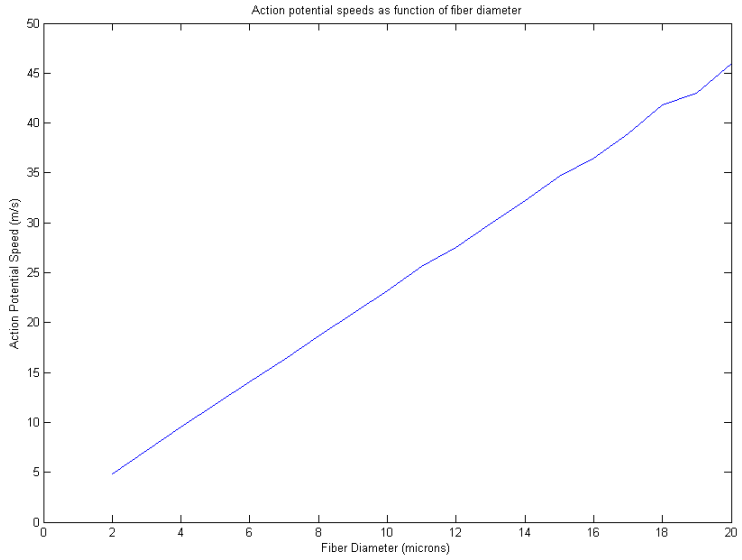


Figure 5.2: Curve showing action potential propagation speed as a function of fiber diameter for the model used in the project.

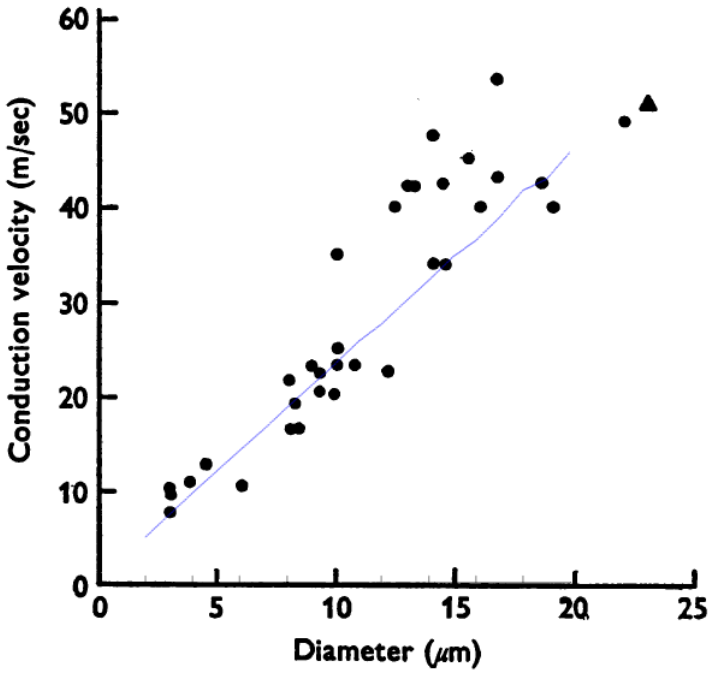


Figure 5.3: Compared simulation and experimental data curves for action potential conduction speed as a function of fiber diameter. Experimental data graph taken from [20]. The simulation curve has been scaled so that the axes on both curves matched.

### 5.2.2 electric field distributions

The second part of the model's characterization involves studying block thresholds and compare values with the literature. Since this is the second, more realistic model, the electric field generated by the blocking electrode has been modelized in COMSOL, with figures 5.4 and 5.5 showing the different distributions for a monopolar electrode (current return at the boundaries of the model) and a bipolar electrode respectively.

In a purely resistive medium with longitudinal resistivity $\rho_l = 300\Omega.cm$ and radial resistivity $\rho_r = 300\Omega.cm$, the electric field wanes very quickly with distance from the electrode, and spreads out significantly, explaining the poor spatial selectivity of cuff electrodes whose electrodes remain on the surface of nerves, resulting in greater distances between the electrodes and the fibers they are supposed to stimulate.

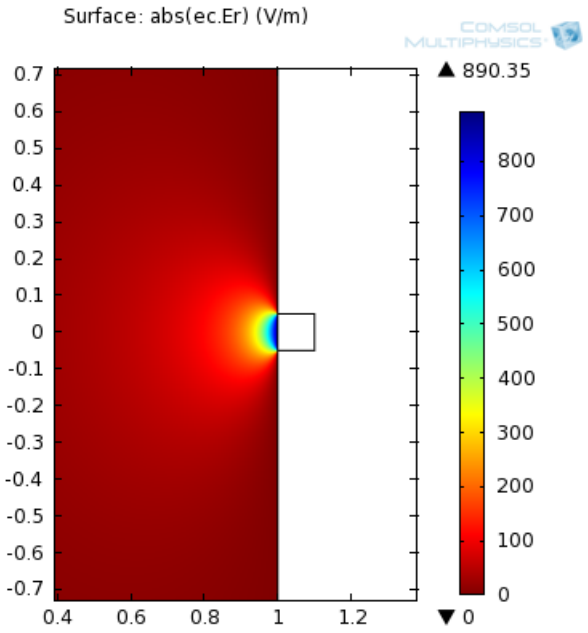


Figure 5.4: Graphical description of the electric field generated by a monopolar electrode in a homogeneous, purely resistive medium.

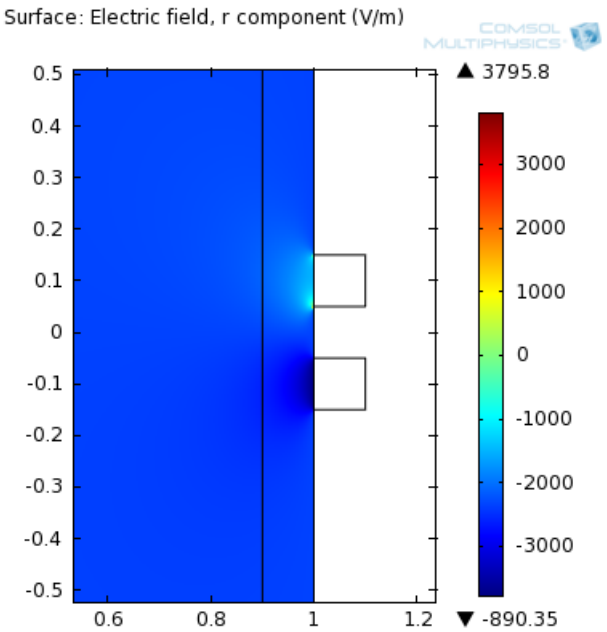


Figure 5.5: Graphical description of the electric field generated by bipolar electrodes in a homogeneous, purely resistive medium.

### 5.2.3 block thresholds

With these values at our disposal, it is now possible to move on to block threshold characterization for our fiber and electrode model.

For characterization of block thresholds, [22] gave AC block thresholds at around 3000-5000 Hz for frequency and at around 0.8 mA for amplitude. As it has been noted in the same paper that increasing the blocking signal frequency also increases the amplitude threshold at which block is achieved, characterization of the block is done over frequency and amplitude ranges appropriate to our situation. Frequency varies from 2500 Hz to 5000 Hz in 100 Hz steps, and amplitude varies from 2.5 mA to 12.5 mA in 0.5 mA steps. This strikes a good balance between simulation time and precision of the results. The uncertainty of the model's predictions are thus $\pm$ 100 Hz and $\pm$ 0.5 mA for frequency and amplitude respectively. The amplitude scale was chosen for 5-micron fibers as smaller fibers have been reported to be harder to block, and fiber diameters in Kilgore and Bhadra's study were in the 12-14 micron range for most.

Since the simulation uses a bruteforce algorithm to test every combination of amplitude and frequency in the ranges specified, the graphs are drawn as 2D plots where every mark denotes a successful block. Unfortunately this method of presentation makes it impossible to combine the block threshold data for different fiber diameters into one graph, and so graphs for 4, 5, 6, 7, 9 and 12 micron fibers are presented[2]. Comparison of the six graphs with each other gives a clear idea of the trends which are detailed in section 6. Respectively the six graphs can be seen fig. 5.6, 5.7, 5.8, 5.9, 5.10 and 5.11.

A cursory inspection of the plots shows that as a general rule, the larger the axon, the easier it is to block it, and that the block depends more on frequency than on blocking signal amplitude, even though there is a visible amplitude threshold especially for smaller (5 micron) fibers. 4-micron fibers are nearly impossible to block according to the model's predictions, for possible reasons outlined in section 6. From the graphs, it is possible to suppose that if the smallest fiber in a simulation is blocked, so will the larger fibers if the blocking signal is of same frequency and amplitude.

What is more striking initially are the multiple points that behave like outliers, showing successful block scenarios where we do not expect any; we would expect for example that for the thresholds in the 5-micron fiber, at around 4400 Hz, looking at the aspect of the area of successful block parameters we would not expect block to be possible at this frequency, yet it is for very low block signal amplitudes. As the modeling is entirely deterministic we cannot attribute this to chance, but perhaps it is a model artefact, for reasons detailed in chapter 6. From inspection of the graphs

[2]These were chosen to show the evolution in block thresholds with fiber diameter more clearly; for example thresholds between 9 and 15 micron fibers do not change noticeably.

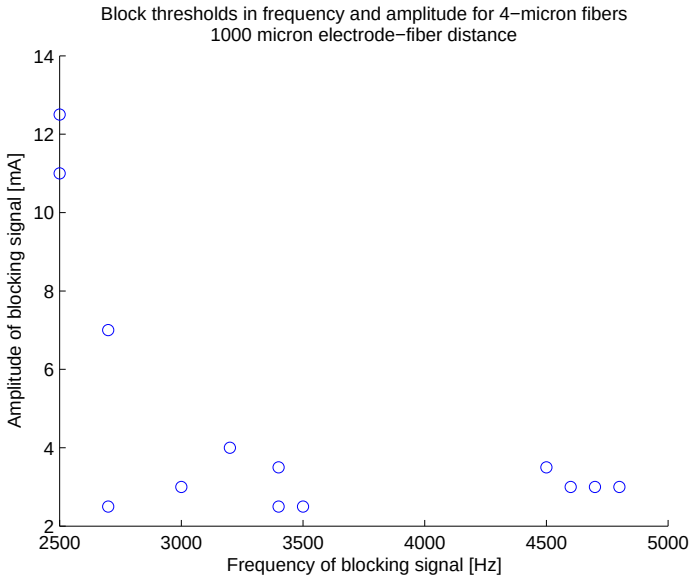


Figure 5.6: Block thresholds in frequency and amplitude for a 4-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

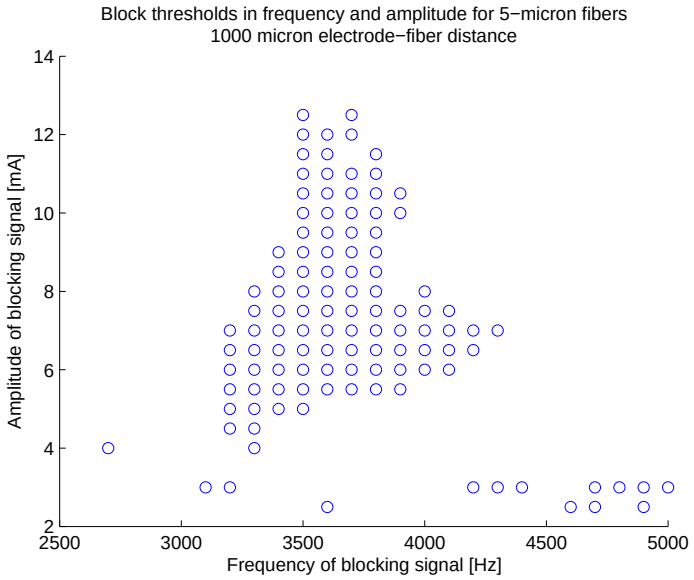


Figure 5.7: Block thresholds in frequency and amplitude for a 5-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

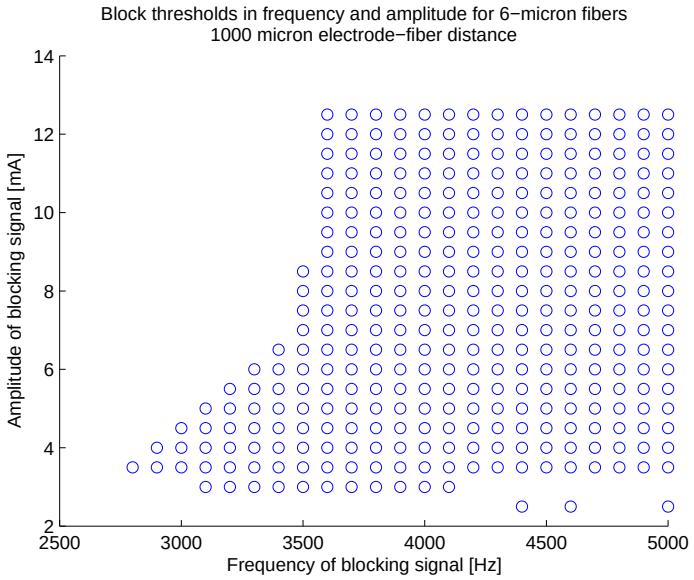


Figure 5.8: Block thresholds in frequency and amplitude for a 6-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

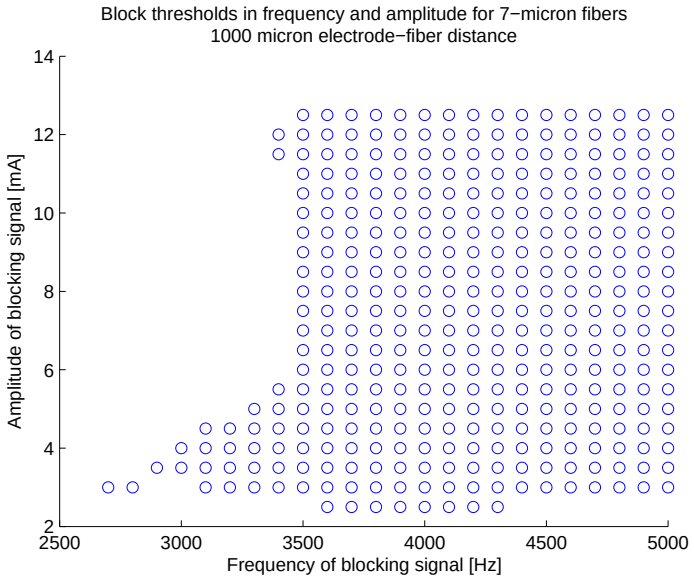


Figure 5.9: Block thresholds in frequency and amplitude for a 7-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

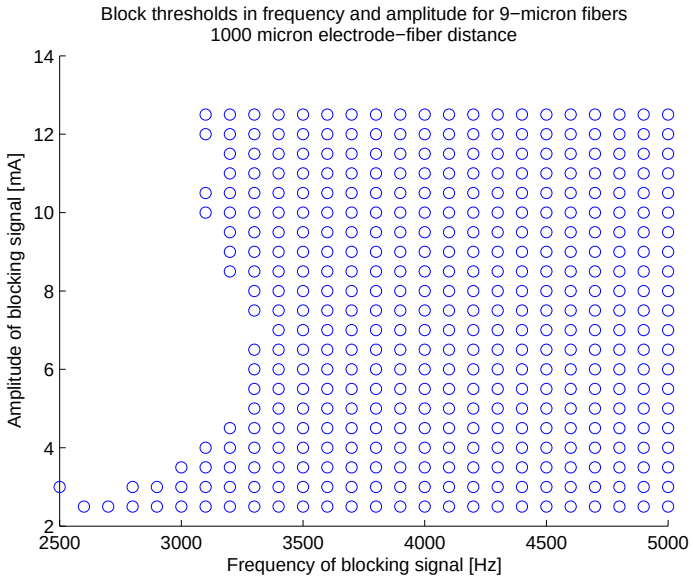


Figure 5.10: Block thresholds in frequency and amplitude for a 9-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

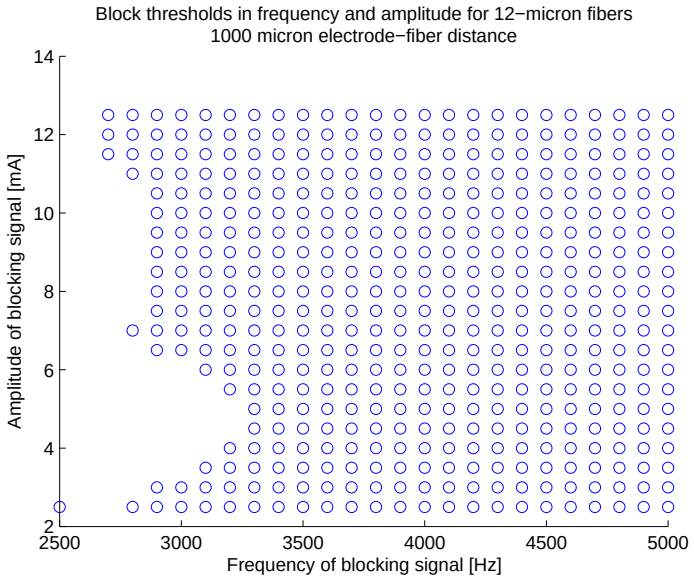


Figure 5.11: Block thresholds in frequency and amplitude for a 12-micron fiber at 1000 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

for fibers of more than 9 microns in diameter, it becomes clear that the threshold depends essentially on frequency as virtually all block amplitudes achieve a block. There was no maximum block amplitude recorded in the tested range for these larger fibers.

In terms of validation with experimental data, comparing the model's predictions to the experimental findings of Kilgore and Bhadra [22], at first the model's predictions for block thresholds may seem high at around 4 mA, 3300 Hz for 5-micron fibers, but the sciatic nerve of *Xenopus Laevis* contains fibers with diameters ranging from 2 to 24 microns with most of the fibers between 12 and 14 microns [20]. Hence since Kilgore and Bhadra's experiments were carried out on this nerve it is likely that their measured thresholds would be lower than that predicted for smaller fibers, i.e 5-micron fibers. Furthermore, since the model has certain simplifications explained in section 6, these may affect predicted block thresholds. In essence, we consider that these results validate the model in terms of block thresholds.

### 5.2.4 block dissipation time

The measured block dissipation times (see table 5.2.4) show a clear trend: at small fiber diameters (5-6 micron ± 1 micron) the dissipation time is slightly lower than for larger diameters, which might be explained by the fact that electrical block at one frequency and amplitude is more effective in larger fibers than in smaller fibers, and thus takes longer to dissipate. The dissipation period used in the selective stimulation protocols was then chosen as 1.4 ms as lower values for smaller fibers would actually ensure that the slower action potential passed after the block in its fiber had dissipated, whereas faster action potentials would always arrive at the blocked area of their fiber before this. Somewhat confusing results were obtained with certain fiber diameters such as the 8-micron fiber where the dissipation period is noticeably smaller than for other fibers both larger and smaller. This means that in certain cases it should be impossible for the protocol to ensure selective stimulation at small fiber diameter differences and interelectrode distances because the difference in TOA of the two action potentials at the blocked area would not be large enough to filter the faster one out. It should be noted that in the literature, refractory periods are reported to

| Fiber diameter [microns] | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|
| Block Dissipation Period [ms] | 1.1 | 1.2 | 1.6 | 1.4 | 1.6 | 1.7 | 1.6 | 1.6 |

Table 5.1: Table for the measured block dissipation times for different fiber diameters. Blocking signal parameters were 4000 Hz, 5.5 V. All measurement uncertainties for fiber diameter and block dissipation period are ± 1 micron and ± 0.1 ms respectively.

decrease with increasing fiber diameter [29].

## 5.3 geometrical constraints of a hypothetical implant using AC timing block

### 5.3.1 interelectrode distance

For a hypothetical implant using the technique, geometrical constraints such as minimum length of the implant and maximum nerve diameter for reliable selective stimulation need to be defined. The results for the relevant simulations mentioned in section 4 are located here.

Observation of the graph leads us to conclude that there is a minimum interelectrode distance of about 8000 microns (8 mm) that is needed in the design of an implant to reliably differentiate between fibers with a diameter difference equal to 2 microns. Note that the protocol's maximum resolution is about 1.5 micron diameter difference at the tested lengths, and that at certain interelectrode distance the protocol isn't able to successfully selectively stimulate the smaller fibers when the diameter difference is smaller than 5 microns, for reasons explained in section 6.

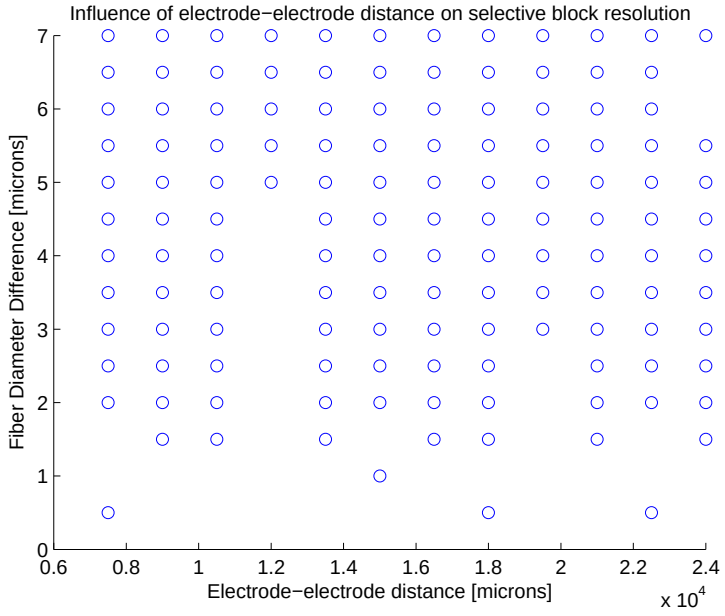


Figure 5.12: Graph representing the influence of stimulating electrode to blocking electrode distance on selective stimulation resolution, with a monopolar blocking electrode.

### 5.3.2 electrode-fiber distance

The nerve diameter for which the technique remains reliable has a maximum on the other hand. This parameter is essentially electrode-fiber distance, and as the electric field quickly spreads out from its source, it is logical that the field intensity reaching the axon quickly wanes as electrode-fiber distance increases. This can be seen fig 5.13 with the distribution of the vertical [3]component of the electric field values over the 30000 micron fiber.

Note the shifted electric field distribution in the case of the bipolar electrode configuration: this is due to how the model identifies the position of the electrode in the comsol file by looking at the maximum electric field values at a certain electrode-fiber distance. Block threshold simulations for the different electrode-fiber distances for the monopolar electrode are displayed below.

The plots indicate that as the fiber is simulated closer to the electrode the threshold plots start ressembling those of larger fibers (see fig. 5.10). Vice-versa for longer electrode-fiber distances where the plot for the block thresholds of a 5-micron fiber 1250 microns from the electrode ressemble those for a 4-micron fiber 1000 microns from the electrode (see fig. 5.6.) The

[3]the vertical component in COMSOL corresponds here to the component of the electric field that is perpendicular to the fiber, which is responsible for the transmembrane current.

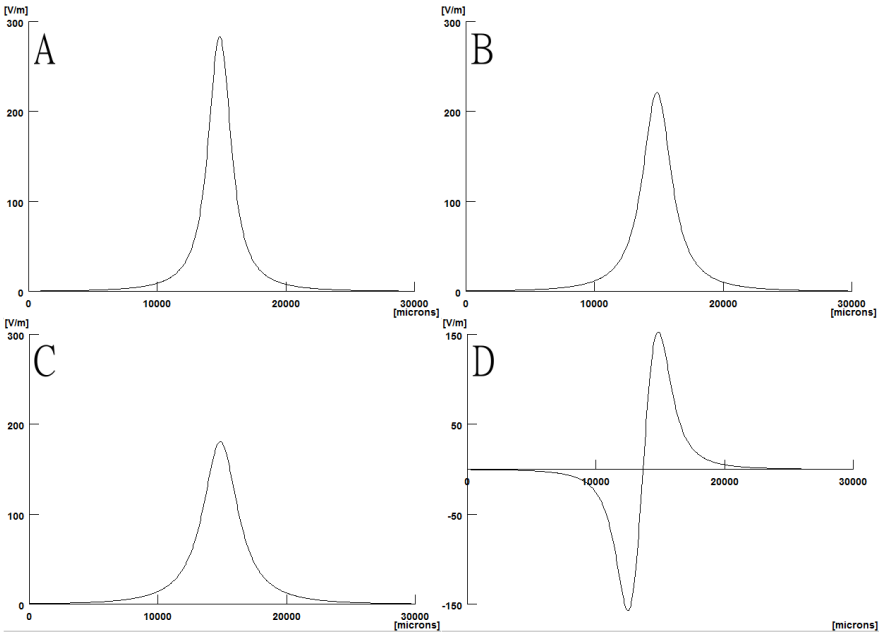


Figure 5.13: Curves for the distribution of the vertical component of the electric field over the axon at several electrode-fiber distances and electrode configurations. A: 750 microns monopolar, B: 1000 microns monopolar, C: 1250 microns monopolar, D: 3000 microns bipolar.

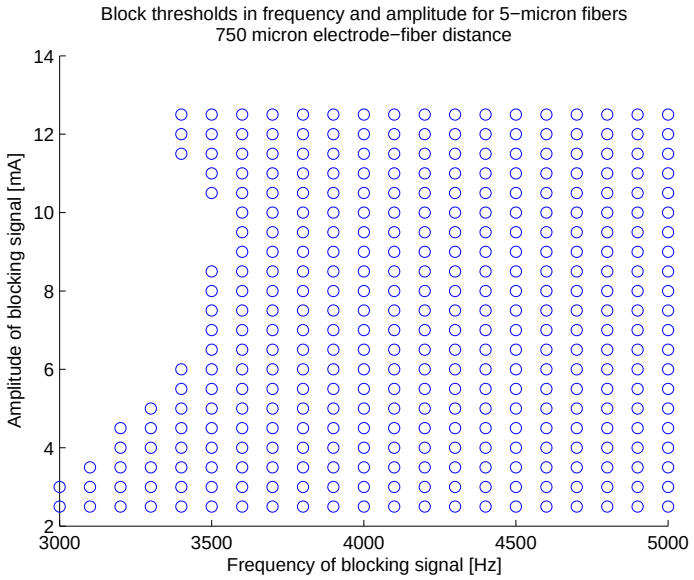


Figure 5.14: Block thresholds in frequency and amplitude for a 5-micron fiber at 750 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

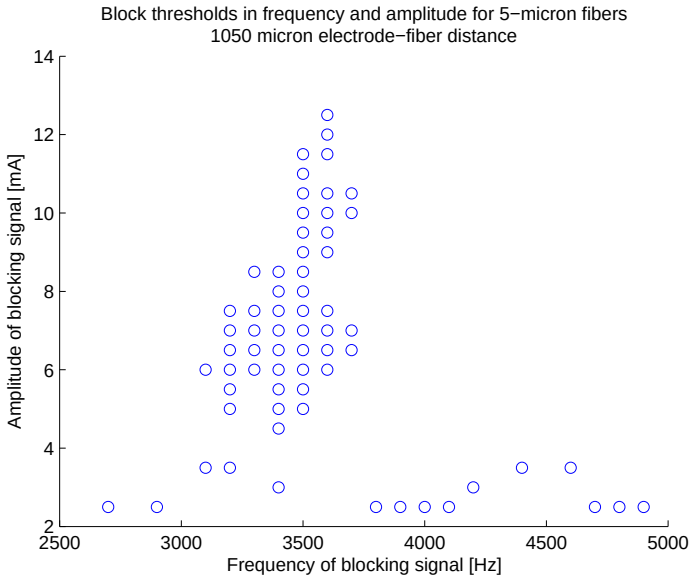


Figure 5.15: Block thresholds in frequency and amplitude for a 5-micron fiber at 1050 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

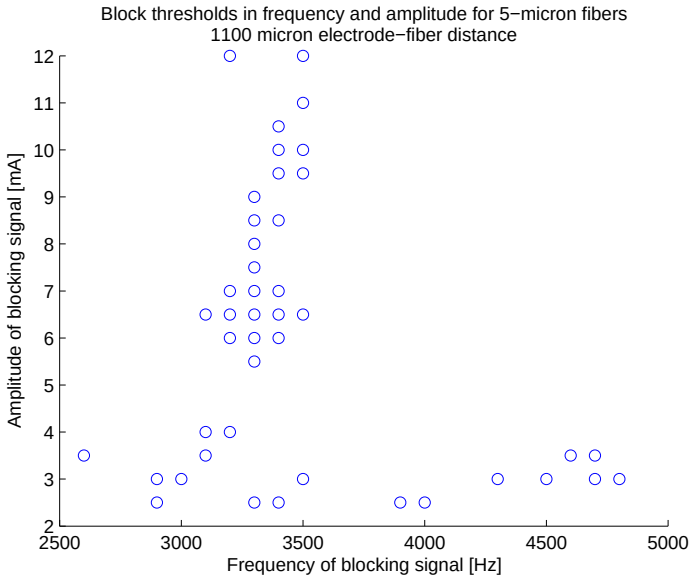


Figure 5.16: Block thresholds in frequency and amplitude for a 5-micron fiber at 1100 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

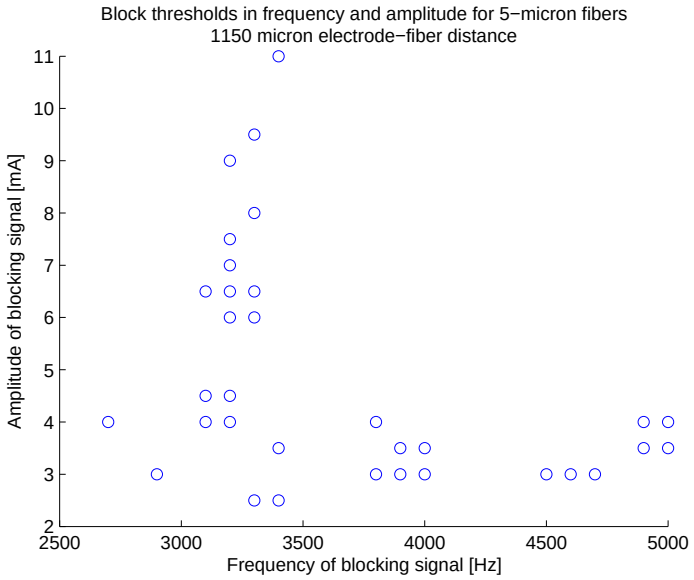


Figure 5.17: Block thresholds in frequency and amplitude for a 5-micron fiber at 1150 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

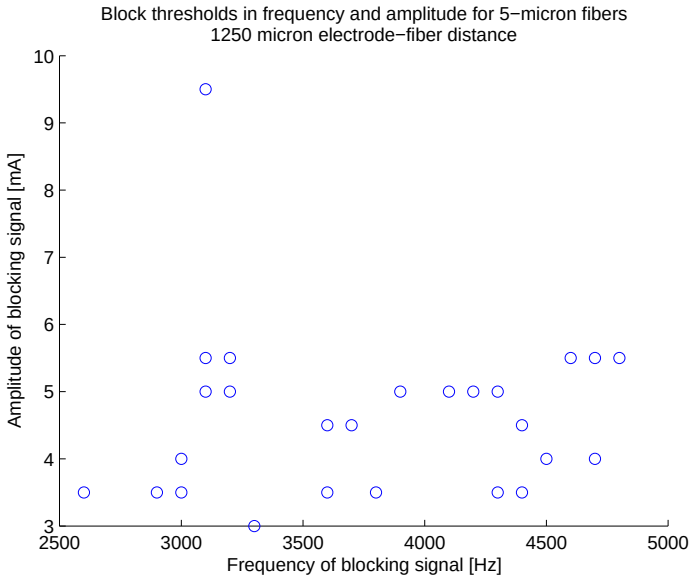


Figure 5.18: Block thresholds in frequency and amplitude for a 5-micron fiber at 1250 microns from a monopolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

results indicate that for fibers with diameters smaller than 5 microns, block becomes difficult or even impossible at 1250 microns from the electrode, according to the model's predictions. Thus it is evident that the ability of an implant using the timing block technique to block all fibers in the implanted nerve is going to depend on electrode-fiber distance. Due to the significantly lower peak values of the electric field distributions, block will only be effective at higher thresholds and yet this causes more problems (see section 6) which result in block failure even at high blocking signal amplitudes as can be seen from the figures. Additional tested electrode-fiber distances (see figures 5.15, 5.16 and 5.17) allow us to determine that the maximum electrode-fiber distance where block is reliable is around 1100-1150 microns for a 5-micron diameter fiber.

## 5.4 results with a bipolar blocking electrode

### 5.4.1 block thresholds

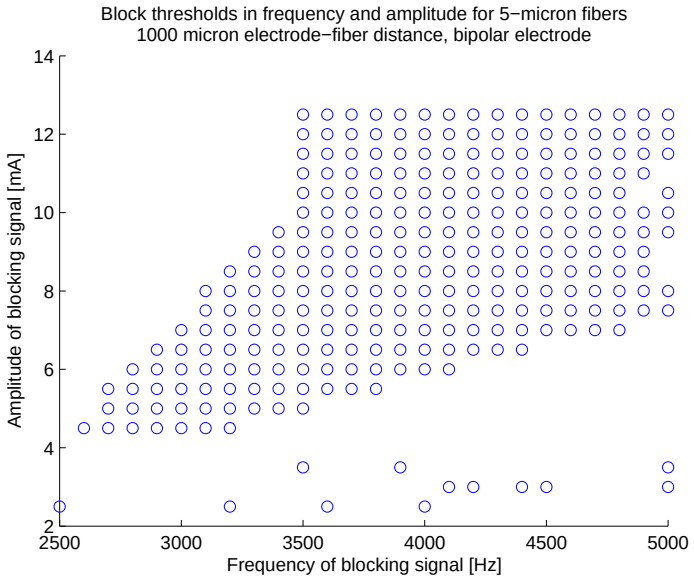


Figure 5.19: Block thresholds in frequency and amplitude for a 5-micron fiber at 1000 microns from a bipolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

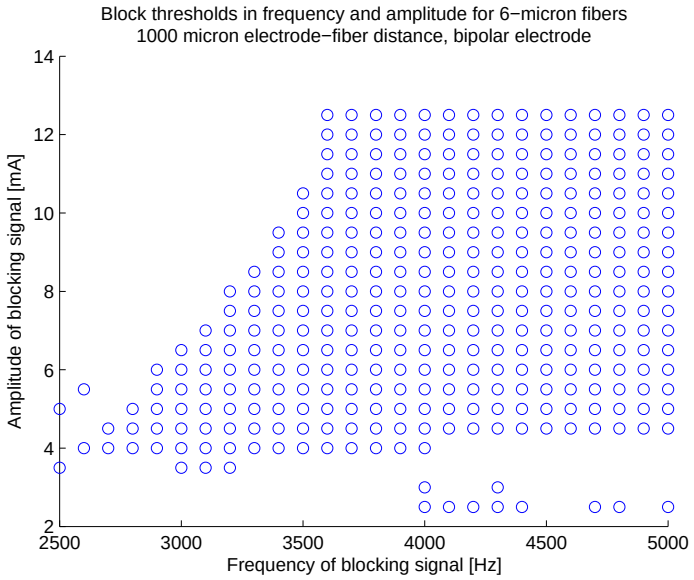


Figure 5.20: Block thresholds in frequency and amplitude for a 6-micron fiber at 1000 microns from a bipolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

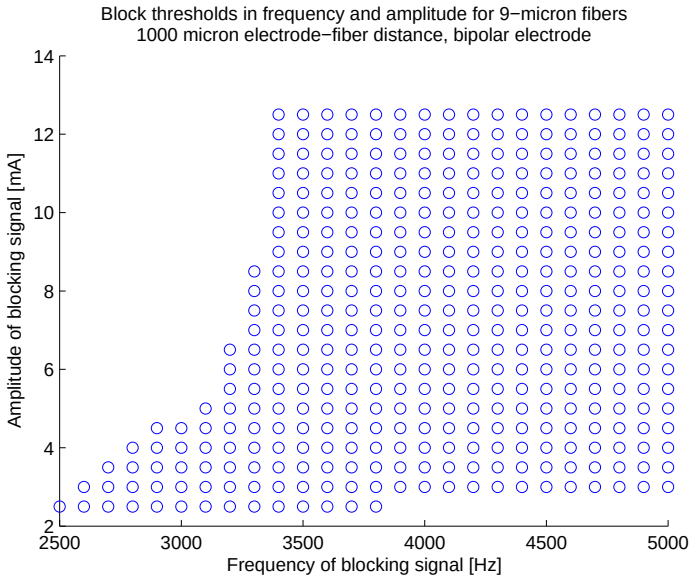


Figure 5.21: Block thresholds in frequency and amplitude for a 9-micron fiber at 1000 microns from a bipolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

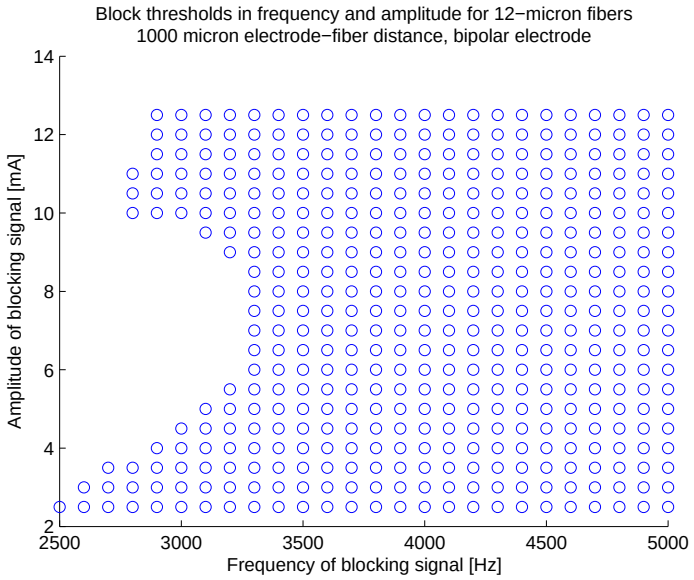


Figure 5.22: Block thresholds in frequency and amplitude for a 12-micron fiber at 1000 microns from a bipolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

### 5.4.2 interelectrode distance

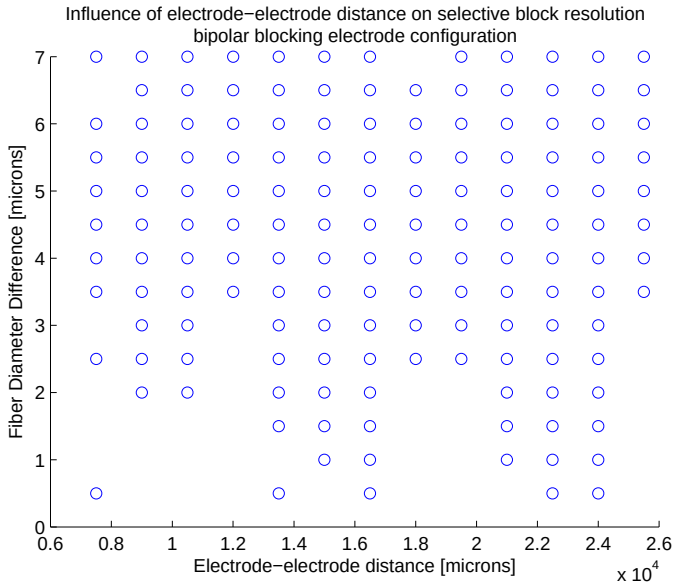


Figure 5.23: Graph representing the influence of stimulating electrode to blocking electrode distance on selective stimulation resolution with a bipolar blocking electrode.

The block thresholds plot clearly shows that bipolar block is in this case more efficient than monopolar block, with a larger spread of successful block parameters for a 5-micron fiber, with explanations in section 6.

Selective stimulation is shown to be possible over the same range of inter-electrode distances but with lower resolution. The minimum distance before reaching maximum resolution is about 17000 microns (17 mm). Similarly to the corresponding graph with a monopolar blocking electrode, at certain interelectrode distances the resolution of the protocol degrades noticeably, though this is probably due to model artefacts (see section 6).

# Chapter 6

# discussion

To discuss the validity of the results obtained during the project, tentative explanations for some of the more unexpected observations are needed.

## 6.1 conduction speeds

On the subject of action potential conduction speeds, the model seems to be agreeing for a large part with the experimental data. While this data does not feature error bars with which a more precise judgement could be made, the curve output by the model clearly stays within the observed ranges for action potential conduction speed in *Xenopus Laevis*, and so we consider the model validated in this regard.

## 6.2 block thresholds

Looking at the results on block thresholds, with the observed trend that smaller fibers are more difficult to block than larger fibers, some more precise analysis can be obtained with the inclusion of the graphs for thresholds of a 5 micron diameter fiber at varying electrode-fiber distances (see figures 5.7, 5.15, 5.16, 5.17 and 5.18). First of all larger fibers are generally easier to block but this isn't always true, for example when comparing block thresholds for 5 and 7 micron diameter fibers at 1000 microns. Indeed in this case, at borderline block conditions such as blocking signal amplitude 6 at frequency 3300 Hz, the block is successful for 5-micron fibers but not for 7-micron fibers. Outside of these borderline conditions, the rule remains true. It can moreover be noticed that there are two frequency thresholds, one at around 3000 Hz when blocks start becoming successful at low blocking signal amplitudes, and another at 3500 Hz when the blocking signal amplitude can be raised significantly higher than at lower frequencies without resulting in a failure of the block.

The model seems to overestimate block thresholds compared to experimental data as Kilgore and Bhadra found block was possible at amplitudes of just 0.8 mA, and the model predicts 3-4 mA for 5-micron fibers. However they did use the sciatic nerve which mostly contains fibers in the ranges of 12-14 microns in diameter which could explain the discrepancies as larger fibers are easier to block. Furthermore, simplifications made by the model compared to reality may explain differences.

For example the model does not represent paranodes, which are elements linking the internodes to the nodes proper. These elements play a role in action potential conduction speed [5, 37], and may play a role in block thresholds as well.

Then, the model represents the fiber with a constant diameter, while in reality the thickness of the myelin defines the fiber's diameter while the nodes of Ranvier have a smaller diameter corresponding to that of the actual axon. This smaller diameter is generally considered to be a constant fraction of the fiber diameter (coefficient 0.7 [28]).

Finally, myelinated sections in the model are considered perfectly insulating, when in reality there is a nonzero myelin conductance. This may affect conduction speed as well.

From the block threshold simulation results for varying electrode-fiber distances, it is apparent that the farther the fiber is from the electrode, the harder it is to block, and this is correlated with the electric field distribution (see figures 5.13) which spreads out with lower peak values. This makes it necessary to use higher blocking signal amplitudes to achieve block at the center of the axon, but this results in action potential generation at the adjacent nodes which defeats the purpose of the block itself. Increasing the blocking signal amplitude has no effect, since it just displaces the stimulated nodes further from the blocked area as lower values in the electric field distribution are able to reach the stimulation threshold of the axon. Additionally, this explains why "sharper" electric field distributions can more easily block small diameter fibers, as less current spreads to adjacent nodes. It is important to note that with the significant block threshold differences between simulations done at electrode-fiber distances of 1000, 1050, 1100 and 1150 microns, the ability of the electrode to block the nerve is extremely sensitive to this distance, and this should be taken into account in a hypothetical implant design to make sure that this distance is known, does not vary, and that it is within the limits for blocking, which depend on the diameter of the smallest fiber to block. In the case that this is a 5-micron fiber, the rule of thumb is not to exceed 1100 microns in electrode-fiber distance, as can be seen from the graphs since this is the longest electrode-fiber distance at which there is a substantial interval of amplitude-frequency parameters for which block is successful (around 3500 Hz, 6.5 mA according to fig. 5.16). For longer distances it is very difficult or impossible finding successful block

parameters valid with all fiber diameters.

The influence of internode length on blocking efficiency could be tested by running block threshold simulations with electrode-fiber distance and internode length as variables: if it is indeed the electric field distribution's spread which affects block, nodes which are more spread out will be more effectively blocked, and so larger axons, or axons with a larger internode length to diameter ratio, will be blocked more effectively at longer fiber-electrode distances. While we have observed that larger axons are easier to block then smaller fibers at longer electrode-fiber distances (see fig. 8.3 in section 8.2), it remains to be tested whether axons with the same diameter, but with different internode length to diameter ratios, are easier to block when the internode distance is longer.

## 6.3 interelectrode distance

For the influence of interelectrode distance on the blocking technique's resolution, first of all it is evident that at the chosen amplitude and frequency for the selective stimulation simulation, block was reliably achieved when the blocking and stimulatory electrodes were far enough appart. Indeed, with interelectrode distances as low as 7500 microns (7.5 mm), the protocol can achieve selective stimulation with fiber diameter differences as low as 2 microns, meaning that with "small" fibers whose diameter ranged from 5 to 10 microns (and 12 microns for the "large" fiber), block was achieved within the 20 milliseconds of onset response time, and was then cut off without any kind of break excitation, while the stimulation pulse was correctly timed, resulting in slective stimulation of the smaller fiber. These results are consistent accross a wide range of interelectrode distances, suggesting that the minimum interelectrode distance should actually be 7.5 mm. However, theoretically better resolution can be obtained the farther the stimulatory and blocking electrodes are from each other as the longer distance to travel for the action potentials before reaching the block coupled with the propagation speed difference results in a larger difference in the TOA, making it easier to time the stimulation pulse. Depending on the application and the fiber size composition of the implanted nerve, an appropriate interelectrode distance can be chosen from graphs similar to fig. 5.12, which can be made more precise with a reduction in the time and spatial steps, as well as using more detailed models in terms of neuroanatomy.

## 6.4 results with the bipolar blocking electrode

As to the effects of a bipolar electrode configuration compared to a monopolar one, it has been reported in the literature that bipolar block is more effective than monopolar block, yet this has not been fully explained. A tentative explanation can be made by noticing that the electric field distribution has a smaller spread per electrode, which can be seen fig 5.13 (subfigure D). Along with results showing that there are a wider range of amplitudes and frequencies for the blocking signal for which block is achieved with a bipolar electrode configuration (compare figures 5.7 and 5.19), this could explain why bipolar block is more effective than monopolar block.

## 6.5 model limitations

However, for both block threshold characterization and selective stimulation simulations it seems that the model suffers from noise of some kind as the results are unreliable at certain parameter ranges, notably low blocking signal amplitudes for blocking thresholds, and certain interelectrode distances for selective stimulation. This can be explained by noticing that the blocking effect is essentially a divergent phenomenon: the block is either active, at which point no amount of incoming pulses of equal amplitude can destabilize it, or it is not, in which case it will spontaneously generate action potentials at variable rates, in situations reported by the literature as partial blocks [22]. The electric field values are assigned directly to the different locations on the nerve, yet the spacial resolution in the COMSOL file is of 50 microns due to computational time constraints both for generation of the file, and extraction of values from it by NEURON. Since nodes of Ranvier are 3 microns long, this leads to some approximation in electric field values. Hence, at borderline parameter values for block success or failure, there is a degree of noise due to quantization, even though all results can be reliably reproduced by executing the same code. Furthermore, there may be additional model artefacts, the effects of which can influence the results of simulations. The interelectrode distance simulations clearly show an unexpected periodic effect: at certain interelectrode distances the ability of the protocol to selectively stimulate the smaller fiber is considerably reduced as successful attempts are only recorded with large fiber diameter differences. In a real nerve experiment, it is expected that as successful selective stimulation was achieved for smaller fiber diameter differences at larger and smaller interelectrode distances, this effect should not be observed.

In essence, while the results obtained during this project show that improvement of selectivity with existing electrodes is possible, some limitations of the modelization and of the technique in general must be kept in mind.

The model does not represent the axons it attempts to replicate in every detail. While it is considered that the deviations from reality have only a comparatively small quantitative effect on the results of the simulations, these simplifications may serve to explain the more confusing results obtained in some plots, for example fig. 5.6. To enumerate some of the axon properties that the model omits:

1. The paranode elements which link the nodes of Ranvier to the internodes are absent in this model when they are present in other models of axons, and they have a role in action potential conduction.

2. The fiber diameter variations from the internode to the node of Ranvier are not taken into account and so the axon's diameter is the fiber diameter at all locations. The main effect is that more node surface area is exposed compared to a more realistic axon, and this may affect conduction speed (though the simulated conduction speed has been validated with experimental data) and blocking thresholds.

3. The myelin sheath is considered here perfectly insulating and this affects how the axon responds to external stimulation, since in reality the myelin has a nonzero conductance. Furthermore, when it was attempted to represent the myelin more realistically in the model by giving it conductance values from [38], abnormal behaviour was observed where the membrane potential of myelinated segments rose with the passage of an action potential (see section 8.2, figures 8.1 and 8.2). This is unexpected and the underlying mechanism has to be understood if a more realistic representation of myelin is to be made.

4. As it stands the model predicts that it isn't possible to selectively stimulate fibers that have diameter differences equal or inferior to 1.5 microns reliably: it is successful in some cases and not in others. As the modelization is entirely deterministic, this may indicate that the model suffers from noise, an aspect which could be improved.

5. Fibrous encapsulation of implanted electrodes and it's influence on the electric field distribution wasn't modeled. While the technique relied on the fact that this fibrous encapsulation could be ignored if certain stimulus parameters were used to obtain block of every fiber in an implanted nerve, this may not be applicable with regards to safety regulations on blocking signal amplitudes.

6. The stimulation electrode remained intracellular, which is unrealistic if the implant is to be based on cuff electrodes. While the stimulation of fibers in the nerve is possible with the cuffs, the eventual interference of the stimulation cuff with the blocking cuff wasn't fully investigated,

and could lead to increased electrode-electrode distances in practical applications. However, for the purposes of knowing whether it is possible to stimulate all fibers in a nerve with an extracellular electrode, this is confirmed with the model itself since all blocks at the extracellular blocking electrode generate at least one action potential before block is established.

7. Most of the results were obtained using a monopolar extracellular electrode model for practical reasons as this made identification of the position of the electrode more straightforward, and timing of the stimulatory pulse for selective stimulation easier. Most if not all implants would use either bipolar or tripolar electrodes, and as initial simulations show that using bipolar electrodes improves block quality with larger bands of successful block parameters, further investigation of the technique using bipolar electrode models is warranted.

8. The electric field values were assigned to their respective locations on the nerve directly: an improvement would be to use interpolation to assign exact values to the segment centers, based on the `node_segnum` and `myelin_segnum` parameters, though the changes in scale from nodes to myelinated sections can make this more complicated.

Furthermore, the timing block technique used in this project to obtain selective stimulation has inherent drawbacks and may not be applicable in every situation. There are several aspects which can be improved:

1. The technique produces an onset action potential every time it is used, and it is supposed that for every stimulated AP that is selectively sent through small fibers, the block must be turned on and then off again, producing an onset action potential every time. It becomes evident that a solution must be found to prevent the onset response from ocurring if this technique is to be used chronically. Work has already been done on this aspect of AC stimulation [2], however the techniques used in the paper haven't been tested in the context of this project.

2. Implants using this technique require a long cuff electrode (more than 3 cm a priori) to be implanted in certain situations where the difference in diameter of the fibers that have to be selectively stimulated is small (less than 2 microns). This may pose problems for implanting nerves where only a small length of the nerve is accessible *in vivo*. In these situations the implantation of two nerve cuffs at different locations may be warranted.

3. The success of the technique depends on precise knowledge of the conduction speeds of the fibers that are to be blocked, and of those that are to be selectively stimulated. Hence if any unexpected change in

conduction speed occurs for the fibers in the implanted nerve, the technique is rendered ineffective. A possible solution would be to include the possibility of measuring conduction speed with the implant alongside its selective stimulation capabilities. With knowledge of the fiber diameters within the nerve, it would then be pssible for the implant to adapt its timings if any change in conduction speed were observed. This could also double as an indicator of fiber health as reductions in conduction speed have been associated with nerve damage in the literature.

# Chapter 7

# conclusion

## 7.1 scientific contribution

Selective stimulation was modeled and shown to be possible in a realistic representation of myelinated axons of *Xenopus Laevis*. Amongst several possible methods for fiber size-selective stimulation of nerve, a timing-based method was identified as the most appropriate. A protocol for selective stimulation using this technique was designed and implemented in the myelinated axon model and shown to work under a variety of conditions with a resolution of 2 microns in fiber diameter difference. The model shows that the technique can work with a monopolar electrode and results suggest that this is also true for the more realistic bipolar electrode configuration. The protocol is robust relative to changes in nerve geometry, cuff electrode geometry and action potential conduction speed changes, provided that these are known. It should be indifferent to the formation of encapsulation tissue, provided that this tissue does not change transfer impedances between the electrode and the nerve enough as to make total nerve block impossible. In this manner appropriate blocking signal parameters can be used so that block is ensured over the time of encapsulation tissue formation, after which electrical properties do not change noticeably.

The following protocol for selective stimulation in the nerves of *Xenopus Laevis* is the result of analysis of the simulations mentioned in this report:

The protocol specifies that with two electrodes, one proximal for stimulation and one distal for blocking:

1. A total nerve block must be established prior to selective stimulation. Depending on the onset response time period this may take several tens of milliseconds. Optimal values for a monopolar blocking electrode are 3500 Hz and 6.5 mA, as this corresponds to successful block values for all tested fiber diameters from 5 to 15 microns at 1000 microns

of electrode-fiber distance, compatible with block thresholds for 5-micron fibers at 1050 electrode-fiber distance. This means that the radius of the largest nerve on which the technique will reliably work is 2100 microns (2.1 mm). A priori the same parameters can be used successfully with a bipolar blocking electrode.

2. With known blocking signal cutoff time, action potential propagation speed, interelectrode distance and refractory period, a stimulatory pulse for the whole nerve is timed so that the faster action potential is annihilated. The timing equation is: $t_{pulse} = t_{bco} + t_{bd} - \frac{t_{ps}+t_{pf}}{2}$. Interelectrode distance must be larger than 7.5 mm between the stimulatory and blocking electrodes.

3. The pulse is sent proximal to the blocked area with the simulatory electrode. According to the model and if the parameters are within the modeled limits, selective stimulation of the smaller fiber is achieved.

The code used for simulations is provided in the appendix of this report and is meant as a base on which improvements can be made in future investigations of this subject. Functions were designed to be modular and easy to use and understand so that they may be modified without excessive debugging.

A user manual has been created and can be used to repeat all of the simulations mentioned in this report, and recreate all the corresponding graphs, as well as carry out further work with models both more realistic and representing nerves with different behaviours, with the eventual goals of applying the technique in humans with spinal cord injury.

## 7.2 suggestions for further work

Validation of the model's predictions, especially with regards to the selective stimulation simulations, would be of great value and can be readily carried out on nerves of *Xenopus Laevis* with the appropriate laboratory equipment, using a setup similar to the one described in this report. Code similar to the one used in the models can be used for stimulation pulse timing purposes, especially if membrane voltage sensors are placed in the same locations as in the model.

Verification that the selective stimulation protocol remains effective for different ion channel models is another logical step to take with this research. However care should be taken in the interpretation of results especially in the absence of experimental data as the model has displayed unexpected behaviour, and it is of interest to investigate why, so that erroneous predictions can be avoided.

# Bibliography


[1] Neri Accornero, GIORGIO Bini, GIAN LUIGI Lenzi, and M Manfredi. Selective activation of peripheral nerve fibre groups of different diameter by triangular shaped stimulus pulses. *The Journal of physiology*, 273 (3):539–560, 1977.

[2] Douglas Michael Ackermann. *Reduction of the onset response in high frequency nerve block.* PhD thesis, Case Western Reserve University, 2010.

[3] D Michael Ackermann Jr, Niloy Bhadra, Emily L Foldes, and Kevin L Kilgore. Separated interface nerve electrode prevents direct current induced nerve damage. *Journal of neuroscience methods*, 201(1):173–176, 2011.

[4] KW Altman and R Plonsey. Analysis of the longitudinal and radial resistivity measurements of the nerve trunk. *Annals of biomedical engineering*, 17(4):313–324, 1989.

[5] Charles F Babbs and Riyi Shi. Subtle paranodal injury slows impulse conduction in a mathematical model of myelinated axons. *PloS one*, 8 (7):e67767, 2013.

[6] Mark F Bear, Barry W Connors, and Michael A Paradiso. *Neuroscience*, volume 2. Lippincott Williams & Wilkins, 2007.

[7] Niloy Bhadra and Kevin L Kilgore. Direct current electrical conduction block of peripheral nerve. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12(3):313–324, 2004.

[8] EA Blair and Joseph Erlanger. A comparison of the characteristics of axons through their individual electrical responses. *American Journal of Physiology–Legacy Content*, 106(3):524–564, 1933.

[9] Almut Branner, Richard B Stein, and Richard A Normann. Selective stimulation of cat sciatic nerve using an array of varying-length microelectrodes. *Journal of neurophysiology*, 85(4):1585–1594, 2001.

[10] Theodore Bullock and G Adrian Horridge. Structure and function in the nervous systems of invertebrates. 1965.

[11] JC Eccles and Charles S Sherrington. Numbers and contraction-values of individual motor-units examined in some muscles of the limb. *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character*, 106(745):326–357, 1930.

[12] Consortium for Spinal Cord Medicine. Bladder management for adults with spinal cord injury: a clinical practice guideline for health-care providers. *The Journal of Spinal Cord Medicine*, 29(5):527–573, 2006.

[13] B Frankenhaeuser and AF Huxley. The action potential in the myelinated nerve fibre of xenopus laevis as computed on the basis of voltage clamp data. *The Journal of Physiology*, 171(2):302–315, 1964.

[14] Meana Gerges, Emily L Foldes, D Michael Ackermann, Narendra Bhadra, Niloy Bhadra, and Kevin L Kilgore. Frequency-and amplitude-transitioned waveforms mitigate the onset response in high-frequency nerve block. *Journal of neural engineering*, 7(6):066003, 2010.

[15] Warren M Grill and J Thomas Mortimer. Electrical properties of implant encapsulation tissue. *Annals of biomedical engineering*, 22(1): 23–33, 1994.

[16] Warren M Grill and J Thomas Mortimer. Stability of the input-output properties of chronically implanted multiple contact nerve cuff stimulating electrodes. *Rehabilitation Engineering, IEEE Transactions on*, 6 (4):364–373, 1998.

[17] Michael Hines. Xenopus myelinated neuron (frankenhaeuser, huxley 1964), January 2002. URL `http://senselab.med.yale.edu/ModelDB/ShowModel.asp?model=3507`.

[18] Michael L Hines and Nicholas T Carnevale. The neuron simulation environment. *Neural computation*, 9(6):1179–1209, 1997.

[19] Michael L Hines and Nicholas T. Carnevale. Expanding neuron's repertoire of mechanisms with nmodl. *Neural Computation*, 12(5):995–1007, 2000.

[20] NA Hutchinson, ZJ Koles, and RS Smith. Conduction velocity in myelinated nerve fibres of xenopus laevis. *The Journal of physiology*, 208(2): 279, 1970.

[21] Laveeta Joseph and Robert J Butera. High-frequency stimulation selectively blocks different types of fibers in frog sciatic nerve. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 19(5): 550–557, 2011.

[22] KL Kilgore and N Bhadra. Nerve conduction block utilising high-frequency alternating current. *Medical and Biological Engineering and Computing*, 42(3):394–406, 2004.

[23] St W Kuffler and RW Gerard. The small-nerve motor system to skeletal muscle. *Journal of neurophysiology*, 10(6):383–394, 1947.

[24] Stephen W Kuffler and EM Vaughan Williams. Small-nerve junctional potentials. the distribution of small motor nerves to frog skeletal muscle, and the membrane characteristics of the fibres they innervate. *The Journal of physiology*, 121(2):289–317, 1953.

[25] GE Loeb and RA Peck. Cuff electrodes for chronic stimulation and recording of peripheral nerve activity. *Journal of neuroscience methods*, 64(1):95–103, 1996.

[26] Donald R McNeal. Analysis of a model for excitation of myelinated nerve. *Biomedical Engineering, IEEE Transactions on*, (4):329–337, 1976.

[27] Daniel R Merrill, Marom Bikson, and John GR Jefferys. Electrical stimulation of excitable tissue: design of efficacious and safe protocols. *Journal of neuroscience methods*, 141(2):171–198, 2005.

[28] John W Moore, Ronald W Joyner, Michael H Brill, Stephen D Waxman, and Manuel Najar-Joa. Simulations of conduction in uniform myelinated fibers. relative sensitivity to changes in nodal and internodal parameters. *Biophysical journal*, 21(2):147–160, 1978.

[29] Robert P Morse and Edward F Evans. The sciatic nerve of the toad¡ i¿ xenopus laevis¡/i¿ as a physiological model of the human cochlear nerve. *Hearing research*, 182(1):97–118, 2003.

[30] Xavier Navarro, Thilo B Krueger, Natalia Lago, Silvestro Micera, Thomas Stieglitz, and Paolo Dario. A critical review of interfaces with the peripheral nervous system for the control of neuroprostheses and hybrid bionic systems. *Journal of the Peripheral Nervous System*, 10 (3):229–258, 2005.

[31] F Rattay. The basic mechanism for the electrical stimulation of the nervous system. *Neuroscience*, 89(2):335–346, 1999.

[32] NJM Rijkhoff, H Wijkstra, PEV Van Kerrebroeck, and FMJ Debruyne. Urinary bladder control by electrical stimulation: review of electrical stimulation techniques in spinal cord injury. *Neurourology and urodynamics*, 16(1):39–53, 1997.

[33] Bradley J Roth. Mechanisms for electrical stimulation of excitable tissue. *Critical reviews in biomedical engineering*, 22(3-4):253–305, 1993.

[34] WAH Rushton. A theory of the effects of fibre size in medullated nerve. *The Journal of physiology*, 115(1):101, 1951.

[35] Moshe Solomonow. External control of the neuromuscular system. *Biomedical Engineering, IEEE Transactions on*, (12):752–763, 1984.

[36] Wale Sulaiman and Tessa Gordon. Neurobiology of peripheral nerve injury, regeneration, and functional recovery: from bench top research to bedside application. *The Ochsner Journal*, 13(1):100–108, 2013.

[37] Wenjing Sun, Yan Fu, Yuzhou Shi, Ji-Xin Cheng, Peng Cao, and Riyi Shi. Paranodal myelin damage after acute stretch in guinea pig spinal cord. *Journal of neurotrauma*, 29(3):611–619, 2012.

[38] Ichiji Tasaki. New measurements of the capacity and the resistance of the myelin sheath and the nodal membrane of the isolated frog nerve fiber. *Am J Physiol*, 181(3):639–650, 1955.

[39] Ichiji Tasaki. A new measurement of action currents developed by single nodes of ranvier. *J. Neurophysiol*, 27:1199–1206, 1964.

[40] Ichiji Tasaki, K Ishii, and H Ito. On the relation between the conduction-rate, the fiber-diameter and the internodal distance of the medullated nerve fiber. *Jpn J Med Sci III, Biophysics*, 9(1):189–99, 1943.

[41] G Theophilidis and PAVLINA Pavlidou. The vitality of the sciatic nerve of the frog and rat in a chamber which allows maintained in vitro recording of the compound nerve action potentials. *Muscle & nerve*, 16 (1):113, 1993.

[42] Dustin J Tyler and Dominique M Durand. Functionally selective peripheral nerve stimulation with a flat interface nerve electrode. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(4): 294–303, 2002.

[43] JG Whitwam and C Kidd. The use of direct current to cause selective block of large fibres in peripheral nerves. *British journal of anaesthesia*, 47(11):1123–1134, 1975.

# Chapter 8

# annex

## 8.1 user's manual for the simulation toolbox

In this section usage instructions are included for those who would use the software toolbox made during this project. All model files are included in one folder, and most files are separate functions and procedures used by the main project files.

The files used by the model can be separated into several categories:

- Main files: these are the files that are run by the user. They contain all the model's parameters except the ones determined in COMSOL simulations and the channel behaviours. They call other functions in order to run the simulations, and execute tests based on the goal of the simulation. Based on the simulation purpose, several versions of a similar base file can be found.
- Function files: these contain the largest part of the code used to run the simulations. They are loaded at startup by the Main files. They are designed to be modular and only require the arguments specified at the beginning of the file to run.
- Data files: these contain data that the Main files require for their simulations as part of the arguments passed to the called functions. They include the electric field distribution, action potential propagation speeds as a function of fiber diameter and other eventual predetermined values.

In order to understand how the model runs a simulation, detailing the command file structure is required.

1. File loading commands: These are at the beginning of Main files and correspond to all the other files used by the Main file to run its programmed simulation.

2. Parameter Space: This section allows the user to specify model parameters such as geometry, stimulation and blocking parameters (frequencies, amplitudes, delays), and other parameters.

3. Reset and Initialization procedures: They are included in the main file for practical purposes as any changes to the names of any variables must also be reflected in these procedures so that changed variables are correctly initialized and reset. The reset procedure destroys and recreates all objects. The initialization procedure makes most function calls in order to build and initialize the model for a new simulation. Both of these procedures are called at the start of a new simulation. Unless new model features are added, they generally do not need to be changed.

4. Command Space: This is the part of the main file that the user can change in order to program a batch of simulations, test the result of each simulation, and plot and export these results into graphs and text files respectively.

Each Main file contains the relevant program structures for a particular type of test. Every Main file contains one or more "for" loops that allow certain model parameters to change from one simulation to the next. Vectors containing the values for these parameters are created at the beginning of the command space, and then these values are used to change model parameters within the for loops, e.g. blocking signal frequency and amplitude for blocking tests. The model is reset first, then parameters are changed, the model is initialized and the simulation is run. The test on the simulation is run, and the result stored in the appropriate structure. At the end of the batch run, the results are output to out.txt, which can be read and plotted by other programs such as MATLAB.

The following includes details on the model parameters. Most of the parameters are self-explanatory, but for the sake of clarity and ease-of-use, descriptions of these parameters follow:

- cvode_active: this parameter determines if NEURON uses the CVode variable time step methods for simulation. These have been found to be much slower than fixed time step methods with negligible differences in results, but have been left should modifications be made to the code that make them more useful for these simulations.

- Temperature: this parameter affects channel dynamics, and from there action potential conduction speed and possibly blocking thresholds (untested). It was left at 20 C to simulate conditions for a whole nerve preparation of *Xenopus Laevis* at which the preparation is maintained at room temperature.

- Segmentation parameters: these are internal to NEURON and set how many compartments will be used to modelize the different sections (nodes of Ranvier and myelinated sections) within the model.[1]

- General Geometry Parameters: these change the diameter, node length and internode length of the axons, and allows the user to set a "desired total length" for the modelled axons as the actual modelled length will be determined by the node and internode lengths.

- Electrode positioning parameters: these can be used to adjust the "position" of the electrode over the modelled fiber(s). The first parameter is the relative x position of the electrode, since the axon length can change from simulation to simulation. The second parameter is the modelled distance from the fiber to the electrode. In both cases this only affects how the COMSOL file containing the electric field values is read and the values assigned to the segments on the axon. The rest of the parameters specify the file from which to read the values, and the minimum resolution of the COMSOL simulation in microns (50 microns is typical, but for general purposes any higher numbe can be used).

- Neurophysiological Parameters: These define the duration of the refractory period and expected onset response for stimulation timing purposes within the simulation.

- Stimulation parameters: these specify how the nerve is stimulated at the proximal (left) end to generate distally-propagating action potentials. Pulses are used for stimulation with configurable amplitude, delay between pulses, duration of pulses, and number of pulses.

- AC Blocking Parameters: blocking signal amplitude, frequency, and starting starting phase can be specified here, with the duration calculated from previously determined parameters, even though it can be specified manually. If specified manually, it can still be slightly modified by other functions during initialization for charge balance: the

[1] As a rule of thumb the "d-lambda" rule is used: nseg must be set so that the segment length is a small fraction of d-lambda, the characteristic length for membrane potential falloff. In the simulations the node was simulated with 1 segment and myelinated sections with 5 as a good balance between precision and simulation speed, verified by plotting results with higher numbers of segments and comparing.

function acvectinit has to be modified in order to prevent this from happening.

- Simulation Parameters: These affect core simulation parameters like the time step, generally set to 0.005 as a good balance between precision and simulation time, even though values like 0.0005 can be used for easier viewing of simulation "movies".[2] The starting time is generally set to 0, and the final time is calculated from other parameters to ensure that no important simulation events are cut off from the run.

- Capacitances and Resistances: these can be used to set the passive electrical properties of the membrane and myelin sheath. In the files provided they are set for frog nerves.

- Object Definitions: these create the objects for the simulations for the first time: the statements are necessary at command level due to how the HOC language behaves.

When initializing the model, several aspects of the model are defined in the following order. Note that the functions and procedures called mulitple times, once for each modeled axon, allow axon properties to be individually tailored in a simulation.

1. Derived geometry parameters such as the length of the internodes are computed from user-input parameters. The number of nodes per axon is calculated and displayed for each modeled axon.

2. Section properties like ion channel mechanism, passive capacitance and axoplasmic resistance are set with calls to specify_sections, once for each axon.

3. Axons are assembled by connecting two half internodes with a node in the middle, and then connecting the half internodes together. This is done with a call to assemble_axons for each axon.

4. The APcounter mechanisms are inserted, allowing for the detection of action potentials passing at a certain node. They should not be inserted at myelinated internodes since they monitor membrane potential and this potential doesn't rise aboce the +30 mV triggering threshold with the passage of action potentials in a perfectly insulating internode model. implant_APcounters is called once for each axon.

[2]Simulation movies refer to some of the graphical output seen when the graphical parameter is set to 1, with plots showing membrane potential over the length of the modelled axon(s) changing with time.

5. The COMSOL-generated file with removed headers is read for the electric field distributions. COMSOLvector is called once for each axon as their lengths are taken into account when initializing the vector containing electric field values.

6. In the relevant simulations, the action potential propagation speeds are read from a text file. This is necessary for selective stimulation simulations.

7. The blocking signal vector is initialized, ensuring charge balance and preventing break excitation if parameters have been set correctly. stimvec_is caled once for each axon.

8. In relevant simulations, the stimulus timing in order to obtain selective stimulation is computed with determine_stim_timing

9. The calculated stimulus timing is used along with user-specified stimulus parameters to initialize the stimulation signal vector. stimvec_init is called once for each axon.

10. The blocking signal is set for each segment of the modeled axons using the electric field values and the blocking signal vector, with NEURON's vector play method. setblock is called once for each axon.

11. The simulation ending time is set based on previously derived parameters.

12. Intracellular current clamps are inserted. These play the stimulation vectors as injected currents. implant_iclamps is called once for each axon.

Once this initialization protocol has been completed, according to instructions within the for loop for in the program, the simulation is run. Graphical output during the simulation can optionally be turned on in parameter space. The program automatically runs the simulations based on the variable parameter vectors specified before the loop. Results for each simulation are evaluated with a custom function at the end of each run, and recorded into a matrix. Once the for loop has finished, this matrix is output along with corresponding variable parameters in an out.txt file.

The default parameters for a simulation which results in selective stimulation of a 5-micron fiber and a block of a 12-micron fiber are as follows:

```
//Temperature
{celsius=20}

//Segmentation
```

```
node_segnum = 1
myelin_segnum = 5

//General Geometry Parameters
axon_diameter_small = 5
axon_diameter_large = 12
axon_length = 30000

diameter_internode_ratio = 100
node_length = 3

//Electrode positioning
electrode_config = 1
electrode_pos_small = 0.5
electrode_pos_large = 0.5
electrode_ypos = 1000
strdef file_string
if (electrode_config == 2) {file_string = "efield_bipolar.txt"
} else {file_string = "efield_monopolar.txt"}
MIN_resolution = 1000

//Neurophysiological Parameters
refractory_dur = 1.6
onset_response_time = 20

//Stimulation parameters
stim_amp = 10
stim_amp_small = 10
stim_amp_large = stim_amp_small*axon_diameter_large/axon_diameter_small

pulse_delay = 5
pulse_dur = 0.15
pulse_num = 1

//AC stimulation (blocking) parameters
amplitude = 5
acfreq = 3300
if (electrode_config == 2) {acphase = PI/2
} else {acphase = 0}
acdur = onset_response_time + 5.0
acres = 1

//Simulation Parameters (Initialization)
graphical = 1
```

```
v_init = -70
timestep = 0.005
tinit = 0
end_delay = 10


//Capacitances
membrane_surfacic_capacitance = 2
myelin_surfacic_capacitance = 0.005

//Resistances
axosplasmic_resistivity = 100
```

Note that this applies to the main file that simulates 2 axons, named `project_work_file.hoc`. The for loops used for batch runs should be removed or set to carry out one simulation, and no parameters should be modified before model initialization. Graphical is set to 1 to provide output, and timestep can be reduced to 0.0005 if the simulation is too fast to observe what is happening.

For COMSOL files, the mesh should be of type mapped and maximum element size 50 microns. The headers of the output file need to be removed before the function within NEURON can read the file, even though it may be modified so that headers may be ignored.

A note on the procedures used for determining successful blocks and selective stimulations:

These are modular "tests" that output a boolean, generally into a matrix so that multiple parameters can be made to change within batch runs, and their influence on the success of the technique readily studied.

For action potential propagation speed measurements (function `ap_speed_measure`), the APcounters at the proximal end and at the middle of the axon are used. The distance between the two is evaluated and together with the time difference between the detection of the stimulated AP at one and the other, the conduction speed is calculated.

For block thresholds (procedure `block_test`): The APcounters at both extremities of the axon are used to determine if block was successful. Both output a vector containing timestamps for action potential detection times. Times before the expected onset response has ended are deleted in the first part of the procedure by creating new vectors without these values. In the second part of the procedure, the size of these vectors are evaluated, and the APcounter on the proximal side of the fiber must have counted 5 (stimulated) action potentials, while the APcounter on the distal side must

have counted none for the test to be successful. Otherwise it is considered a failure and a code is output on the terminal consisting of 2 numbers: these correspond to the number of detected APs at the proximal and distal end of the simulated axon respectively.

For selective stimulation tests (procedures `success_test` and `success_test_bipolar`), the APcounters at both extremities of the axon are used to determine if selective stimulation is successful. For both the monopolar and bipolar versions, the first part of the procedure is very similar to the test for block thresholds as action potentials generated suring the onset response are ignored. Then the size of each timestamp vector is tested and while both timestamp vectors for the small axon must have a size of one (there was only one stimulated AP and it passed through the blocked area), the proximal and distal APcounters of the large axon must have timestamp vectors of size 1 and 0 respectively (there was only one stimulated AP and it did not pass through the block). There is a further test in the monopolar version where the times at which the stimulated AP was detected are compared for the small axon APcounters. The time difference is then checked against an expected difference calculated from the axon potential conduction speed and the calculated distance between the two APcounters. If the difference between the expected and measured time differences is within 1 ms, then the test is successful, else it is onsidered a failure. This is to ensure that the AP considered to be the one stimulated, isn't confused with one that could have potentially been generated at the blocked area in borderline block equations. This test had to be removed for bipolar blocking electrode configurations as it gave too many false negatives when selective stimulation had actually been achieved.

## 8.2 additional results

It can be seen from the comparison of the two timecourses that the model predicts a rise in the membrane potential of myelinated sections when more

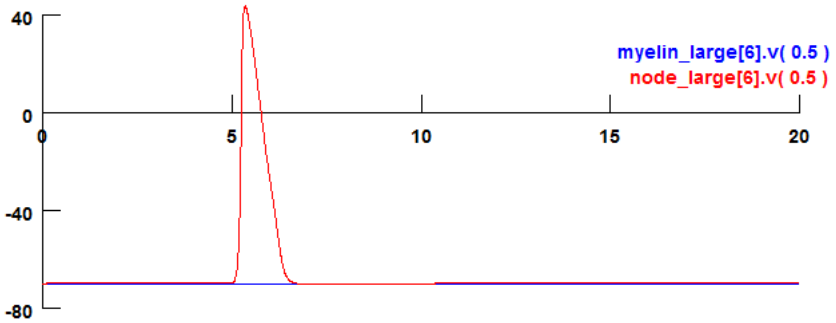


Figure 8.1: Timecourse showing normal membrane voltage behaviour during the passage of an action potential, with perfectly insulating myelin.

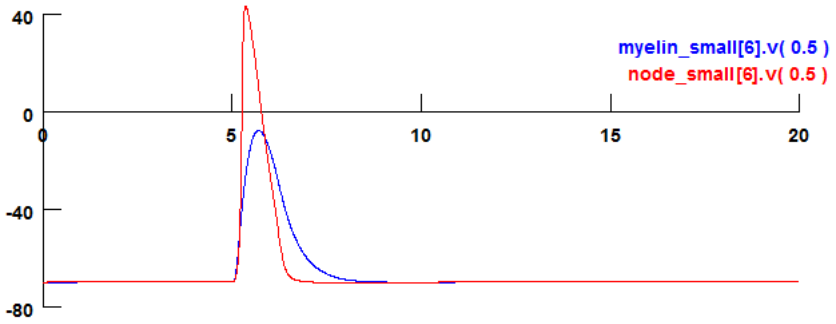


Figure 8.2: Timecourse showing abnormal membrane voltage behaviour during the passage of an action potential, with realistic myelin conductance.

realistic myelin conductance values are used from [38]. This is not expected and points towards possible inconsistencies in the model, that would have to be corrected should the model be made more realistic for future work.

This block threshold graph fig. 8.3 shows that the 9-micron fiber is almost unaffected by greater electrode-fiber distance than 1000 microns while block thresholds of a 5-micron fiber are significantly affected, confirming the hypothesis that if a small fiber can be blocked with a certain set of parameters it is most likely that larger fibers will be blocked at those same parameters.

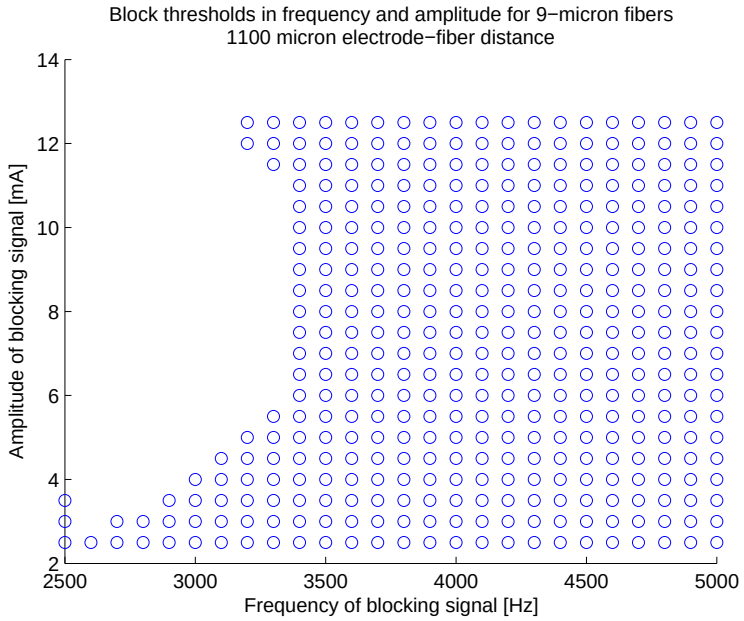


Figure 8.3: Block thresholds in frequency and amplitude for a 9-micron fiber at 1000 microns from a bipolar electrode. Tested frequencies: 2500 to 5000 Hz in 100 Hz steps, tested amplitudes: 2.5 mA to 12.5 mA in 0.5 mA steps.

## 8.3 program code

This section deals with the code used to carry out simulations. All 4 Main files are displayed here first, then the files which are dynamically loaded upon their execution, in order of their loading.

`project_work_file`: This Main file is for general purpose dual axon modeling.

```
1
2    //Adrien Rapeaux Imperial College Bioengineering 4th Year Individual Project
3   load_file("nrngui.hoc")
4   load_file("specify_sections.hoc")
5   load_file("assemble_axons.hoc")
6   load_file("stimvec_init.hoc")
7   load_file("implant_iclamps.hoc")
8   load_file("implant_apcounters.hoc")
9   load_file("acvectinit.hoc")
10  load_file("setblock.hoc")
11  load_file("comsolvector.hoc")
12  load_file("plot_vec.hoc")
13  load_file("vectormovie.hoc")
14  load_file("close_all.hoc")
15  load_file("log_vec.hoc")
16  load_file("lin_vec.hoc")
17  load_file("identify_success_intervals.hoc")
18  load_file("read_action_potential_speeds.hoc")
19  load_file("determine_stim_timing.hoc")
20  load_file("determine_stim_timing_2.hoc")
21  load_file("runcontrol.hoc")
22  load_file("success_test.hoc")
23  load_file("success_test_bipolar.hoc")
24
```

```
25 //Integration Method: calling cvode.active() uses variable time step integration - Warning
       extremely slow
26 //cvode_active(1)
27
28 //Temperature
29 {celsius=20}
30
31 //Segmentation
32 node_segnum = 1   //Number of segments per node
33 myelin_segnum = 5  //Number of segments per half internode
34
35 //General Geometry Parameters
36 axon_diameter_small = 5 //(microns)diameter of small axon fibers in the frog ventral roots [1]
37 axon_diameter_large = 12 //(microns)diameter of large axon fibers in the frog ventral roots
       [1]
38 axon_length = 30000 //microns (this is actually a desired (MAX) length and not a fixed length
       due to fixed length internodes)
39
40 diameter_internode_ratio = 100 //ratio of axon diameter (microns) to internode length (microns
       )
41 node_length = 3 //microns (this doesn't depend on axon diameter)
42
43 //Electrode positioning
44 electrode_config = 1  //2 for bipolar, anything else for monopolar)
45 electrode_pos_small = 0.5 //Position of electrode on axon as percent of total axon length
46 electrode_pos_large = 0.5
47 electrode_ypos = 1000 //microns distance from axon to electrode
48 strdef file_string
49 if (electrode_config == 2) {file_string = "efield_bipolar.txt"  //Electric field data for
       extracellular, bipolar electrode_version
50 } else {file_string = "efield_monopolar.txt"} //Electric field data for extracellular,
       monopolar electrode version
51 MIN_resolution = 1000   //We suppose that the electric field vector values are mapped at least
        every 1000 microns
52
53 //Neurophysiological Parameters
54 refractory_dur = 1.6    //[ms] duration of the refractory period (about 2ms in the frog)
55 onset_response_time = 20  //[ms] duration of the expected onset response to block (empirical
       value)
56
57 //Stimulation parameters
58 stim_amp = 10
59 stim_amp_small = 10
60 stim_amp_large = stim_amp_small*axon_diameter_large/axon_diameter_small
61           //Intracellular stimulation amplitude must be adjusted for different fiber diameters
       .
62           //More volume means more charge needed for the same level of stimulation.
63 pulse_delay = 5   //Delay between each stimulatory pulse [ms]
64 pulse_dur = 0.15  //Stimulation duration 0.15 ms
65 pulse_num = 1   //Number of action potentials to generate
66
67 //AC stimulation (blocking) parameters
68 amplitude = 5  //mA
69 acfreq = 3300 //Hz
70 if (electrode_config == 2) {acphase = PI/2
71 } else {acphase = 0}  //[rad] Serves to avoid break excitation in the model, making detection
       of successful block more straightforward.
72 acdur = onset_response_time + 5.0   //[ms] wait for block stabilization, then start selective
       stimulation protocol
73 acres = 1   //Resolution of acvector [percent] (1 = MAX)
74
75 //Simulation Parameters (Initialization)
76 graphical = 1 //This boolean determines whether the simulation uses graphics; useful for
       diagnostics
77 v_init = -70 //mV
78 timestep = 0.005  //ms
79 tinit = 0
80 end_delay = 10    //[ms] time waited at end of simulation for return to resting state
81
82
83 //Capacitances
84 membrane_surfacic_capacitance = 2     // uF/cm
85 myelin_surfacic_capacitance = 0.005   // uF/cm
86
87 //Resistances
88 myelin_resistivity = 29         //MOhms.cm [2]
89 myelin_axial_resistance_small = myelin_resistivity/(PI*axon_diameter_small^2*((1/0.8)^2-1)
       *(10^-8))   //MOhms/cm calculated from resistivity and myelin cross-sectional area (
       assumed axon diameter to myelin diamter ratio: 0.75)
90 myelin_axial_resistance_large = myelin_resistivity/(PI*axon_diameter_large^2*((1/0.8)^2-1)
       *(10^-8))   //MOhms/cm same as above
91 myelin_surfacic_conductance = 1/(10^5)  // mho/cm
92 axosplasmic_resistivity = 100     //Ohm.cm from standard set of parameters, need additional
       data to confirm for frog neurons
```

```
93  membrane_axial_resistance = 2e14    //Ohms/cm [2]
94
95  //Object Definitions (necessary at command level)
96  objref node_small_ref, myelin_small_ref, node_large_ref, myelin_large_ref, stim_small,
        stim_large, stimvec_small, stimvec_large, apcount_small, apcount_large,
        apcount_timevec_list_small, apcount_timevec_list_large, ap_speed_vect,
        fiber_diameter_vect, acvect, ac_small, ac_large, veclist_small, veclist_large,
        filevect_position, filevect_value, vposition_small, vvalue_small, vposition_large,
        vvalue_large, window_list, rangevarplot_list, window_manager, var_vec_1, var_vec_2,
        result_mat
97  create node_small[1], node_large[1], myelin_small[1], myelin_large[1]
98
99
100
101 //--------------------PROCEDURE SPACE----------------------
102
103
104
105 proc reset_sim() {
106 //Redefine objects to clear existing objects
107 objref node_small_ref, myelin_small_ref, node_large_ref, myelin_large_ref, stim_small,
        stim_large, stimvec_small, stimvec_large, apcount_small, apcount_large,
        apcount_timevec_list_small, apcount_timevec_list_large, ap_speed_vect,
        fiber_diameter_vect, acvect, ac_small, ac_large, veclist_small, veclist_large,
        filevect_position, filevect_value, vposition_small, vvalue_small, vposition_large,
        vvalue_large, window_list, rangevarplot_list, window_manager
108 create node_small[1], node_large[1], myelin_small[1], myelin_large[1]
109 access node_small[0]    //Define default section
110 window_list = new List()
111 rangevarplot_list = new List()
112 window_manager = new PWManager()
113
114 }
115
116 proc init_sim() {local i
117
118   //initialize simulation
119   myelin_length_small = axon_diameter_small*diameter_internode_ratio/2  //Myelin sections are
        defined in half-internodes
120   nb_node_small = int(axon_length/(node_length+axon_diameter_small*diameter_internode_ratio))
121   print "the axon contains ", nb_node_small, "nodes"
122   total_length_small = nb_node_small*(node_length+2*myelin_length_small)
123   print "the total length of the 'small' axon is ", total_length_small, "microns"
124
125   myelin_length_large = axon_diameter_large*diameter_internode_ratio/2  //Myelin sections are
        defined in half-internodes
126   nb_node_large = int(axon_length/(node_length+axon_diameter_large*diameter_internode_ratio))
127   print "the axon contains ", nb_node_large, "nodes"
128   total_length_large = nb_node_large*(node_length+2*myelin_length_large)
129   print "the total length of the 'large' axon is ", total_length_large, "microns"
130
131   create node_small[nb_node_small], node_large[nb_node_large], myelin_small[2*nb_node_small],
        myelin_large[2*nb_node_large]
132   // SectionReference lists must be made at command level since sections cannot be
133   //directly passed as variables
134   node_small_ref = new List()
135   node_large_ref = new List()
136   myelin_small_ref = new List()
137   myelin_large_ref = new List()
138   //Fill SectionReference Lists
139   for i=0, nb_node_small-1 {
140     node_small[i] node_small_ref.append(new SectionRef())
141     myelin_small[2*i] myelin_small_ref.append(new SectionRef())
142     myelin_small[2*i+1] myelin_small_ref.append(new SectionRef())
143     }
144   for i=0, nb_node_large-1{
145     node_large[i] node_large_ref.append(new SectionRef())
146     myelin_large[2*i] myelin_large_ref.append(new SectionRef())
147     myelin_large[2*i+1] myelin_large_ref.append(new SectionRef())
148     }
149   specify_sections(node_small_ref, myelin_small_ref, node_segnum, myelin_segnum,
        axon_diameter_small, node_length, myelin_length_small, membrane_surfacic_capacitance,
        myelin_surfacic_capacitance, axosplasmic_resistivity, myelin_axial_resistance_small,
        membrane_axial_resistance)
150   specify_sections(node_large_ref, myelin_large_ref, node_segnum, myelin_segnum,
        axon_diameter_large, node_length, myelin_length_large, membrane_surfacic_capacitance,
        myelin_surfacic_capacitance, axosplasmic_resistivity, myelin_axial_resistance_large,
        membrane_axial_resistance)
151   assemble_axons(nb_node_small, node_small_ref, myelin_small_ref)
152   assemble_axons(nb_node_large, node_large_ref, myelin_large_ref)
153   implant_APcounters(node_small_ref, nb_node_small, apcount_small, apcount_timevec_list_small)
154   implant_APcounters(node_large_ref, nb_node_large, apcount_large, apcount_timevec_list_large)
155   COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos_small,
        total_length_small, MIN_resolution, vposition_small, vvalue_small, electrode_ypos,
```

```
        node_length, myelin_length_small)
156     COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos_large,
           total_length_large, MIN_resolution, vposition_large, vvalue_large, electrode_ypos,
           node_length, myelin_length_large)
157     read_action_potential_speeds("ap_speeds.txt", ap_speed_vect, fiber_diameter_vect)
158     adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
            ac_small)
159     adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
            ac_large)
160     //stim_delay = determine_stim_timing(ap_speed_vect, fiber_diameter_vect, axon_diameter_small
           , axon_diameter_large,0,electrode_pos_small*(nb_node_small-1),0,electrode_pos_large*(
           nb_node_large-1),node_length+axon_diameter_small*diameter_internode_ratio,node_length+
           axon_diameter_large*diameter_internode_ratio,refractory_dur,adjusted_block_time)
161     stim_delay = determine_stim_timing_2(ap_speed_vect, fiber_diameter_vect, axon_diameter_small
           , axon_diameter_large,total_length_small,electrode_pos_small,total_length_large,
           electrode_pos_large,node_length+axon_diameter_small*diameter_internode_ratio,node_length
           +axon_diameter_large*diameter_internode_ratio,refractory_dur,adjusted_block_time)
162     stimvec_init(stim_delay, timestep, pulse_delay, pulse_dur, pulse_num, stimvec_small)
163     stimvec_init(stim_delay, timestep, pulse_delay, pulse_dur, pulse_num, stimvec_large)
164     setblock(veclist_small, myelin_small_ref, ac_small, myelin_length_small, MIN_resolution,
           vposition_small, vvalue_small, node_small_ref, nb_node_small, node_length, timestep,
           acres)
165     setblock(veclist_large, myelin_large_ref, ac_large, myelin_length_large, MIN_resolution,
           vposition_large, vvalue_large, node_large_ref, nb_node_large, node_length, timestep,
           acres)
166     tfinal = stim_delay+pulse_num*(pulse_dur+pulse_delay)+end_delay //Time for the stimulation,
           including initial and final delays, and time needed for stimulation pulse train.
167     implant_iclamps(node_small_ref, 0, stim_small, stimvec_small, stim_amp, axon_diameter_small,
            timestep, tfinal)
168     implant_iclamps(node_large_ref, 0, stim_large, stimvec_large, stim_amp, axon_diameter_large,
            timestep, tfinal)
169
170     init()
171     if (graphical == 1) {
172       runcontrol(v_init, tinit, tfinal, timestep, 1197, 262)
173     } else {
174       v_init = -70 //mV
175       tstop = tfinal
176       dt = timestep
177       t = tinit
178     }
179
180   }
181
182
183
184   // ------------------------COMMAND SPACE-------------------
185
186
187
188   //Define variable vectors containing increments (might need function in future)
189
190   lin_vec(var_vec_1,1,0.05,0.4)
191   lin_vec(var_vec_2,1,0.5,8)
192   result_mat = new Matrix(var_vec_1.size(),var_vec_2.size())
193
194   //Begin multiple simulation for loop
195   for i=0, var_vec_1.size()-1 {
196     for j=0, var_vec_2.size()-1 {
197
198       //Reset all variables
199       reset_sim()
200
201       // Insert variable parameters here
202       electrode_pos_small = var_vec_1.x(i)
203       electrode_pos_large = var_vec_1.x(i)
204       axon_diameter_small = var_vec_2.x(j)
205
206       //Init&Run simulation
207       init_sim()
208       if (graphical ==1) {
209         plot_vec(filevect_value,window_list,50,50,250,150)
210         plot_vec(acvect,window_list,50,450,250,150)
211         //plot_vec(stimvec_small,window_list,50,450,250,150)
212         vectormovie(myelin_small_ref, axon_length, window_list, rangevarplot_list
           ,550,50,250,150)
213         vectormovie(myelin_large_ref, axon_length, window_list, rangevarplot_list
           ,550,450,250,150)
214       }
215
216       doNotify()
217       run()
218       close_all(window_manager)
219
```

```
220       //Simulation test functions here
221       if (electrode_config == 2) {success_test_bipolar(apcount_timevec_list_small,
           apcount_timevec_list_large, result_mat, var_vec_1, var_vec_2, i, j, onset_response_time)
            //Electric field data for extracellular, bipolar electrode_version
222       } else {success_test(apcount_timevec_list_small, apcount_timevec_list_large, result_mat,
           var_vec_1, var_vec_2, i, j, ap_speed_vect, fiber_diameter_vect, axon_diameter_small, 0,
           nb_node_small-1, node_length+axon_diameter_small*diameter_internode_ratio,
           onset_response_time)}
223
224
225
226       //End for loop here
227     }
228   }
229
230   //Plot Results
231   identify_success_intervals(result_mat,var_vec_2,var_vec_1,window_list,50,50,250,150)
232
233   //Write data
234     //Initialize Write
235     objref f2
236     f2 = new File()
237     f2.wopen("out.txt")
238     //Header
239     f2.printf("NEURON simulation data\n")
240     f2.printf("Model Parameters:\n")
241     f2.printf("Variable 1: from %f to %f\n",var_vec_1.x[0],var_vec_1.x[var_vec_1.size()-1])
242     f2.printf("Variable 2: from %f to %f\n",var_vec_2.x[0],var_vec_2.x[var_vec_2.size()-1])
243     f2.printf("Vector lengths:\n")
244     f2.printf("%f\t%f\n",var_vec_1.size(),var_vec_2.size())
245
246     //Data
247     for i=0, var_vec_1.size()-1 {
248       for j=0, var_vec_2.size()-1 {
249         f2.printf("%f\t%f\t%f\n",var_vec_1.x[i],var_vec_2.x[j],result_mat.x[i][j])
250       }
251     }
252     //Close
253     f2.close()
254
255   /*Sources:
256   [1]: Small nerve junctional potentials. [...] Kuffler and Vaughan Williams
257   */
```

`project_work_file_eedist`: version of the main project file made for characterization of the influence of interelectrode distance and fiber diameter difference on selective stimulation efficiency.

```
 1
 2   //Adrien Rapeaux Imperial College Bioengineering 4th Year Individual Project
 3   load_file("nrngui.hoc")
 4   load_file("specify_sections.hoc")
 5   load_file("assemble_axons.hoc")
 6   load_file("stimvec_init.hoc")
 7   load_file("implant_iclamps.hoc")
 8   load_file("implant_apcounters.hoc")
 9   load_file("acvectinit.hoc")
10   load_file("setblock.hoc")
11   load_file("comsolvector.hoc")
12   load_file("plot_vec.hoc")
13   load_file("vectormovie.hoc")
14   load_file("close_all.hoc")
15   load_file("log_vec.hoc")
16   load_file("lin_vec.hoc")
17   load_file("identify_success_intervals.hoc")
18   load_file("read_action_potential_speeds.hoc")
19   load_file("determine_stim_timing.hoc")
20   load_file("determine_stim_timing_2.hoc")
21   load_file("runcontrol.hoc")
22   load_file("success_test.hoc")
23   load_file("success_test_bipolar.hoc")
24
25   //Integration Method: calling cvode.active() uses variable time step integration - Warning
          extremely slow
26   //cvode_active(1)
27
28   //Temperature
29   {celsius=20}
30
31   //Segmentation
32   node_segnum = 1   //Number of segments per node
33   myelin_segnum = 5  //Number of segments per half internode
```

```
34
35  //General Geometry Parameters
36  axon_diameter_small = 5 //(microns)diameter of small axon fibers in the frog ventral roots [1]
37  axon_diameter_large = 12 //(microns)diameter of large axon fibers in the frog ventral roots
        [1]
38  axon_length = 30000 //microns (this is actually a desired (MAX) length and not a fixed length
        due to fixed length internodes)
39
40  diameter_internode_ratio = 100 //ratio of axon diameter (microns) to internode length (microns
        )
41  node_length = 3 //microns (this doesn't depend on axon diameter)
42
43  //Electrode positioning
44  electrode_config = 2  //2 for bipolar, anything else for monopolar)
45  electrode_pos_small = 0.5 //Position of electrode on axon as percent of total axon length
46  electrode_pos_large = 0.5
47  electrode_ypos = 1000 //microns distance from axon to electrode
48  strdef file_string
49  if (electrode_config == 2) {file_string = "efield_bipolar.txt"  //Electric field data for
        extracellular, bipolar electrode_version
50  } else {file_string = "efield_monopolar.txt"} //Electric field data for extracellular,
        monopolar electrode version
51  MIN_resolution = 1000   //We suppose that the electric field vector values are mapped at least
         every 1000 microns
52
53  //Neurophysiological Parameters
54  refractory_dur = 1.6    //[ms] duration of the refractory period (about 2ms in the frog)
55  onset_response_time = 20  //[ms] duration of the expected onset response to block (empirical
        value)
56
57  //Stimulation parameters
58  stim_amp = 10
59  stim_amp_small = 10
60  stim_amp_large = stim_amp_small*axon_diameter_large/axon_diameter_small
61            //Intracellular stimulation amplitude must be adjusted for different fiber diameters
        .
62            //More volume means more charge needed for the same level of stimulation.
63  pulse_delay = 5   //Delay between each stimulatory pulse [ms]
64  pulse_dur = 0.15  //Stimulation duration 0.15 ms
65  pulse_num = 1   //Number of action potentials to generate
66
67  //AC stimulation (blocking) parameters
68  amplitude = 5  //mA
69  acfreq = 3300 //Hz
70  if (electrode_config == 2) {acphase = PI/2
71  } else {acphase = 0}  //[rad] Serves to avoid break excitation in the model, making detection
        of successful block more straightforward.
72  acdur = onset_response_time + 5.0   //[ms] wait for block stabilization, then start selective
        stimulation protocol
73  acres = 1   //Resolution of acvector [percent] (1 = MAX)
74
75  //Simulation Parameters (Initialization)
76  graphical = 1 //This boolean determines whether the simulation uses graphics; useful for
        diagnostics
77  v_init = -70 //mV
78  timestep = 0.005  //ms
79  tinit = 0
80  end_delay = 10    //[ms] time waited at end of simulation for return to resting state
81
82  //Capacitances
83  membrane_surfacic_capacitance = 2     // uF/cm
84  myelin_surfacic_capacitance = 0.005   // uF/cm
85
86  //Resistances
87  myelin_resistivity = 29         //MOhms.cm [2]
88  myelin_axial_resistance_small = myelin_resistivity/(PI*axon_diameter_small^2*((1/0.8)^2-1)
        *(10^-8))   //MOhms/cm calculated from resistivity and myelin cross-sectional area (
        assumed axon diameter to myelin diamter ratio: 0.75)
89  myelin_axial_resistance_large = myelin_resistivity/(PI*axon_diameter_large^2*((1/0.8)^2-1)
        *(10^-8))   //MOhms/cm same as above
90  myelin_surfacic_conductance = 1/(10^5)  // mho/cm
91  axosplasmic_resistivity = 100     //Ohm.cm from standard set of parameters, need additional
        data to confirm for frog neurons
92  membrane_axial_resistance = 2e14    //Ohms/cm [2]
93
94  //Object Definitions (necessary at command level)
95  objref node_small_ref, myelin_small_ref, node_large_ref, myelin_large_ref, stim_small,
        stim_large, stimvec_small, stimvec_large, apcount_small, apcount_large,
        apcount_timevec_list_small, apcount_timevec_list_large, ap_speed_vect,
        fiber_diameter_vect, acvect, ac_small, ac_large, veclist_small, veclist_large,
        filevect_position, filevect_value, vposition_small, vvalue_small, vposition_large,
        vvalue_large, window_list, rangevarplot_list, window_manager, var_vec_1, var_vec_2,
        result_mat
96  create node_small[1], node_large[1], myelin_small[1], myelin_large[1]
```

```
97
98
99
100 //--------------------PROCEDURE SPACE----------------------
101
102
103
104 proc reset_sim() {
105 //Redefine objects to clear existing objects
106 objref node_small_ref, myelin_small_ref, node_large_ref, myelin_large_ref, stim_small,
        stim_large, stimvec_small, stimvec_large, apcount_small, apcount_large,
        apcount_timevec_list_small, apcount_timevec_list_large, ap_speed_vect,
        fiber_diameter_vect, acvect, ac_small, ac_large, veclist_small, veclist_large,
        filevect_position, filevect_value, vposition_small, vvalue_small, vposition_large,
        vvalue_large, window_list, rangevarplot_list, window_manager
107 create node_small[1], node_large[1], myelin_small[1], myelin_large[1]
108 access node_small[0]    //Define default section
109 window_list = new List()
110 rangevarplot_list = new List()
111 window_manager = new PWManager()
112
113 }
114
115 proc init_sim() {local i
116
117   //initialize simulation
118   myelin_length_small = axon_diameter_small*diameter_internode_ratio/2  //Myelin sections are
        defined in half-internodes
119   nb_node_small = int(axon_length/(node_length+axon_diameter_small*diameter_internode_ratio))
120   print "the axon contains ", nb_node_small, "nodes"
121   total_length_small = nb_node_small*(node_length+2*myelin_length_small)
122   print "the total length of the 'small' axon is ", total_length_small, "microns"
123
124   myelin_length_large = axon_diameter_large*diameter_internode_ratio/2  //Myelin sections are
        defined in half-internodes
125   nb_node_large = int(axon_length/(node_length+axon_diameter_large*diameter_internode_ratio))
126   print "the axon contains ", nb_node_large, "nodes"
127   total_length_large = nb_node_large*(node_length+2*myelin_length_large)
128   print "the total length of the 'large' axon is ", total_length_large, "microns"
129
130   create node_small[nb_node_small], node_large[nb_node_large], myelin_small[2*nb_node_small],
        myelin_large[2*nb_node_large]
131   // SectionReference lists must be made at command level since sections cannot be
132   //directly passed as variables
133   node_small_ref = new List()
134   node_large_ref = new List()
135   myelin_small_ref = new List()
136   myelin_large_ref = new List()
137   //Fill SectionReference Lists
138   for i=0, nb_node_small-1 {
139     node_small[i] node_small_ref.append(new SectionRef())
140     myelin_small[2*i] myelin_small_ref.append(new SectionRef())
141     myelin_small[2*i+1] myelin_small_ref.append(new SectionRef())
142     }
143   for i=0, nb_node_large-1{
144     node_large[i] node_large_ref.append(new SectionRef())
145     myelin_large[2*i] myelin_large_ref.append(new SectionRef())
146     myelin_large[2*i+1] myelin_large_ref.append(new SectionRef())
147     }
148   specify_sections(node_small_ref, myelin_small_ref, node_segnum, myelin_segnum,
        axon_diameter_small, node_length, myelin_length_small, membrane_surfacic_capacitance,
        myelin_surfacic_capacitance, axosplasmic_resistivity, myelin_axial_resistance_small,
        membrane_axial_resistance)
149   specify_sections(node_large_ref, myelin_large_ref, node_segnum, myelin_segnum,
        axon_diameter_large, node_length, myelin_length_large, membrane_surfacic_capacitance,
        myelin_surfacic_capacitance, axosplasmic_resistivity, myelin_axial_resistance_large,
        membrane_axial_resistance)
150   assemble_axons(nb_node_small, node_small_ref, myelin_small_ref)
151   assemble_axons(nb_node_large, node_large_ref, myelin_large_ref)
152   implant_APcounters(node_small_ref, nb_node_small, apcount_small, apcount_timevec_list_small)
153   implant_APcounters(node_large_ref, nb_node_large, apcount_large, apcount_timevec_list_large)
154   COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos_small,
        total_length_small, MIN_resolution, vposition_small, vvalue_small, electrode_ypos,
        node_length, myelin_length_small)
155   COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos_large,
        total_length_large, MIN_resolution, vposition_large, vvalue_large, electrode_ypos,
        node_length, myelin_length_large)
156   read_action_potential_speeds("ap_speeds.txt", ap_speed_vect, fiber_diameter_vect)
157   adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
        ac_small)
158   adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
        ac_large)
159   //stim_delay = determine_stim_timing(ap_speed_vect, fiber_diameter_vect, axon_diameter_small
        , axon_diameter_large,0,electrode_pos_small*(nb_node_small-1),0,electrode_pos_large*(
```

```
        nb_node_large-1),node_length+axon_diameter_small*diameter_internode_ratio,node_length+
        axon_diameter_large*diameter_internode_ratio,refractory_dur,adjusted_block_time)
160   stim_delay = determine_stim_timing_2(ap_speed_vect, fiber_diameter_vect, axon_diameter_small
        , axon_diameter_large,total_length_small,electrode_pos_small,total_length_large,
        electrode_pos_large,node_length+axon_diameter_small*diameter_internode_ratio,node_length
        +axon_diameter_large*diameter_internode_ratio,refractory_dur,adjusted_block_time)
161   stimvec_init(stim_delay, timestep, pulse_delay, pulse_dur, pulse_num, stimvec_small)
162   stimvec_init(stim_delay, timestep, pulse_delay, pulse_dur, pulse_num, stimvec_large)
163   setblock(veclist_small, myelin_small_ref, ac_small, myelin_length_small, MIN_resolution,
        vposition_small, vvalue_small, node_small_ref, nb_node_small, node_length, timestep,
        acres)
164   setblock(veclist_large, myelin_large_ref, ac_large, myelin_length_large, MIN_resolution,
        vposition_large, vvalue_large, node_large_ref, nb_node_large, node_length, timestep,
        acres)
165   tfinal = stim_delay+pulse_num*(pulse_dur+pulse_delay)+end_delay //Time for the stimulation,
        including initial and final delays, and time needed for stimulation pulse train.
166   implant_iclamps(node_small_ref, 0, stim_small, stimvec_small, stim_amp, axon_diameter_small,
       timestep, tfinal)
167   implant_iclamps(node_large_ref, 0, stim_large, stimvec_large, stim_amp, axon_diameter_large,
       timestep, tfinal)
168
169   init()
170   if (graphical == 1) {
171     runcontrol(v_init, tinit, tfinal, timestep, 1197, 262)
172   } else {
173     v_init = -70 //mV
174     tstop = tfinal
175     dt = timestep
176     t = tinit
177   }
178
179 }
180
181
182
183 // ------------------------COMMAND SPACE--------------------
184
185
186
187 //Define variable vectors containing increments (might need function in future)
188
189 lin_vec(var_vec_1,15,0.05,0.15)
190 lin_vec(var_vec_2,15,0.5,5)
191 result_mat = new Matrix(var_vec_1.size(),var_vec_2.size())
192
193 //Begin multiple simulation for loop
194 for i=0, var_vec_1.size()-1 {
195   for j=0, var_vec_2.size()-1 {
196
197     //Reset all variables
198     reset_sim()
199
200     // Insert variable parameters here
201     electrode_pos_small = var_vec_1.x(i)
202     electrode_pos_large = var_vec_1.x(i)
203     axon_diameter_small = var_vec_2.x(j)
204
205     //Init&Run simulation
206     init_sim()
207     if (graphical ==1) {
208       plot_vec(filevect_value,window_list,50,50,250,150)
209       plot_vec(stimvec_small,window_list,50,450,250,150)
210       //plot_vec(acvect,window_list,50,450,250,150)
211       vectormovie(myelin_small_ref, axon_length, window_list, rangevarplot_list
        ,550,50,250,150)
212       vectormovie(myelin_large_ref, axon_length, window_list, rangevarplot_list
        ,550,450,250,150)
213     }
214
215     doNotify()
216     run()
217     close_all(window_manager)
218
219     //Simulation test functions here
220     if (electrode_config == 2) {success_test_bipolar(apcount_timevec_list_small,
        apcount_timevec_list_large, result_mat, var_vec_1, var_vec_2, i, j, onset_response_time)
         //Electric field data for extracellular, bipolar electrode_version
221     } else {success_test(apcount_timevec_list_small, apcount_timevec_list_large, result_mat,
        var_vec_1, var_vec_2, i, j, ap_speed_vect, fiber_diameter_vect, axon_diameter_small, 0,
        nb_node_small-1, node_length+axon_diameter_small*diameter_internode_ratio,
        onset_response_time)}
222
223
224
```

```
225
226       //End for loop here
227     }
228   }
229
230   //Plot Results
231   identify_success_intervals(result_mat,var_vec_2,var_vec_1,window_list,50,50,250,150)
232
233   //Write data
234     //Initialize Write
235     objref f2
236     f2 = new File()
237     f2.wopen("out.txt")
238     //Header
239     f2.printf("NEURON simulation data\n")
240     f2.printf("Model Parameters:\n")
241     f2.printf("Electrode-Electrode distance: from %f to %f [microns]\n",var_vec_1.x[0]*
          axon_length,var_vec_1.x[var_vec_1.size()-1]*axon_length)
242     f2.printf("Fiber Diameter Difference: from %f to %f [microns]\n",axon_diameter_large-
          var_vec_2.x[0],axon_diameter_large-var_vec_2.x[var_vec_2.size()-1])
243     f2.printf("Vector lengths:\n")
244     f2.printf("%f\t%f\n",var_vec_1.size(),var_vec_2.size())
245
246     //Data
247     for i=0, var_vec_1.size()-1 {
248       for j=0, var_vec_2.size()-1 {
249         f2.printf("%f\t%f\t%f\n",var_vec_1.x[i]*axon_length,axon_diameter_large-var_vec_2.x[j],
          result_mat.x[i][j])
250       }
251     }
252     //Close
253     f2.close()
254
255   /*Sources:
256   [1]: Small nerve junctional potentials. [...] Kuffler and Vaughan Williams
257   */
```

`project_work_file_block`: the main file used to characterize block thresholds. Only simulates one axon.

```
 1
 2  //Adrien Rapeaux Imperial College Bioengineering 4th Year Individual Project
 3  load_file("nrngui.hoc")
 4  load_file("specify_sections.hoc")
 5  load_file("assemble_axons.hoc")
 6  load_file("stimvec_init.hoc")
 7  load_file("implant_iclamps.hoc")
 8  load_file("implant_apcounters.hoc")
 9  load_file("acvectinit.hoc")
10  load_file("setblock.hoc")
11  load_file("comsolvector.hoc")
12  load_file("plot_vec.hoc")
13  load_file("vectormovie.hoc")
14  load_file("close_all.hoc")
15  load_file("log_vec.hoc")
16  load_file("lin_vec.hoc")
17  load_file("identify_success_intervals.hoc")
18  load_file("read_action_potential_speeds.hoc")
19  load_file("runcontrol.hoc")
20  load_file("block_test_singleaxon.hoc")
21
22  //Integration Method: calling cvode.active() uses variable time step integration - Warning
        extremely slow
23  //cvode_active(1)
24  //Temperature
25  {celsius=20}
26
27  //Segmentation
28  node_segnum = 1   //Number of segments per node
29  myelin_segnum = 5  //Number of segments per half internode
30
31  //General Geometry Parameters
32  axon_diameter = 5 //(microns)diameter of small axon fibers in the frog ventral roots [1]
33  axon_length = 30000 //microns (this is actually a desired (MAX) length and not a set length
        due to the fixed length internodes)
34
35  diameter_internode_ratio = 100 //ratio of axon diameter (microns) to internode length (microns
        )
36  node_length = 3 //microns (this doesn't depend on axon diameter)
37
38  //Electrode positioning
39  electrode_config = 1  //2 for bipolar, anything else for monopolar)
40  electrode_pos = 0.5   //Position of electrode on axon as percent of total axon length
```

```
41  electrode_ypos = 1000   //microns distance from axon to electrode
42  strdef file_string
43  if (electrode_config == 2) {file_string = "efield_bipolar.txt"  //Electric field data for
        extracellular, bipolar electrode_version
44  } else {file_string = "efield_monopolar.txt"} //Electric field data for extracellular,
        monopolar electrode version
45  MIN_resolution = 1000   //We suppose that the electric field vector values are mapped at least
         every 1000 microns
46
47  //Neurophysiological Parameters
48  refractory_dur = 1.4    //[ms] duration of the refractory period (about 2ms in the frog)
49  onset_response_time = 20  //[ms] duration of the expected onset response to block (empirical
        value)
50
51  //Stimulation parameters
52  stim_amp = 10 //[nA]
53
54  pulse_delay = 5   //Delay between each stimulatory pulse [ms]
55  pulse_dur = 0.15  //Stimulation duration 0.15 ms
56  pulse_num = 5   //Number of action potentials to generate
57
58  //AC stimulation (blocking) parameters
59  amplitude = 5  //mA
60  acfreq = 4000 //Hz
61  if (electrode_config == 2) {acphase = PI/2
62  } else {acphase = 0}  //[rad] Serves to avoid break excitation in the model, making detection
        of successful block more straightforward.
63  acdur = onset_response_time+pulse_num*(pulse_dur+pulse_delay) //ms make sure to block all APs
        to test block for an axon.
64  acres = 1   //Resolution of acvector [percent] (1 = MAX)
65
66  //Simulation Parameters (Initialization)
67  graphical = 0 //This boolean determines whether the simulation uses graphics (1 for active);
        useful for diagnostics
68  v_init = -70 //mV
69  timestep = 0.005  //ms
70  tinit = 0
71  end_delay = 10    //[ms] time waited at end of simulation for return to resting state
72  tfinal = onset_response_time + pulse_num*(pulse_dur+pulse_delay) + end_delay
73
74  //Capacitances
75  membrane_surfacic_capacitance = 2     // uF/cm
76  myelin_surfacic_capacitance = 0.005   // uF/cm
77
78  //Resistances
79  myelin_resistivity = 29         //MOhms.cm [2]
80  myelin_axial_resistance = myelin_resistivity/(PI*axon_diameter^2*((1/0.8)^2-1)*(10^-8))   //
        MOhms/cm calculated from resistivity and myelin cross-sectional area (assumed axon
        diameter to myelin diamter ratio: 0.75)
81  myelin_surfacic_conductance = 1/(10^5)  // mho/cm
82  axosplasmic_resistivity = 100     //Ohm.cm from standard set of parameters, need additional
        data to confirm for frog neurons
83  membrane_axial_resistance = 2e14    //Ohms/cm [2]
84
85  //Object Definitions (necessary at command level)
86  objref node_ref, myelin_ref, stimulator, stimvec, apcount, apcount_timevec_list, ap_speed_vect
        , fiber_diameter_vect, acvect, block_signal, veclist, filevect_position, filevect_value,
         vposition, vvalue, window_list, rangevarplot_list, window_manager, var_vec_1, var_vec_2
        , result_mat
87  create node[1], myelin[1]
88
89
90
91  //--------------------PROCEDURE SPACE----------------------
92
93
94
95  proc reset_sim() {
96  //Redefine objects to clear existing objects
97  objref node_ref, myelin_ref, stimulator, stimvec, apcount, apcount_timevec_list, ap_speed_vect
        , fiber_diameter_vect, acvect, block_signal, veclist, filevect_position, filevect_value,
         vposition, vvalue, window_list, rangevarplot_list, window_manager
98  create node[1], myelin[1]
99  access node[0]    //Define default section
100 window_list = new List()
101 rangevarplot_list = new List()
102 window_manager = new PWManager()
103
104 }
105
106 proc init_sim() {local i
107
108   //initialize simulation
109   myelin_length = axon_diameter*diameter_internode_ratio/2  //Myelin sections are defined in
```

```
        half-internodes
110     nb_node = int(axon_length/(node_length+axon_diameter*diameter_internode_ratio))
111     print "the axon contains ", nb_node, "nodes"
112     total_length = nb_node*(node_length+2*myelin_length)
113     print "the total length of the axon is ", total_length, "microns"
114     create node[nb_node], myelin[2*nb_node]
115     // SectionReference lists must be made at command level since sections cannot be
116     //directly passed as variables
117     node_ref = new List()
118     myelin_ref = new List()
119     //Fill SectionReference Lists
120     for i=0, nb_node-1 {
121       node[i] node_ref.append(new SectionRef())
122       myelin[2*i] myelin_ref.append(new SectionRef())
123       myelin[2*i+1] myelin_ref.append(new SectionRef())
124       }
125
126     specify_sections(node_ref, myelin_ref, node_segnum, myelin_segnum, axon_diameter,
          node_length, myelin_length, membrane_surfacic_capacitance, myelin_surfacic_capacitance,
          axosplasmic_resistivity, myelin_axial_resistance, membrane_axial_resistance)
127     assemble_axons(nb_node, node_ref, myelin_ref)
128     stimvec_init(onset_response_time, timestep, pulse_delay, pulse_dur, pulse_num, stimvec)
129     implant_iclamps(node_ref, 0, stimulator, stimvec, stim_amp, axon_diameter, timestep, tfinal)
130     implant_APcounters(node_ref, nb_node, apcount, apcount_timevec_list)
131     COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos, total_length,
          MIN_resolution, vposition, vvalue, electrode_ypos, node_length, myelin_length)
132     read_action_potential_speeds("ap_speeds.txt", ap_speed_vect, fiber_diameter_vect)
133     adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
            block_signal)
134     setblock(veclist, myelin_ref, block_signal, myelin_length, MIN_resolution, vposition, vvalue
          , node_ref, nb_node, node_length, timestep, acres)
135
136     init()
137     if (graphical == 1) {
138       runcontrol(v_init, tinit, tfinal, timestep, 1197, 262)
139     } else {
140       v_init = -70 //mV
141       tstop = tfinal
142       dt = timestep
143       t = tinit
144     }
145   }
146
147
148
149   // ------------------------COMMAND SPACE--------------------
150
151
152   //Define variable vectors containing increments (might need function in future)
153
154   lin_vec(var_vec_1,26,100,2500)
155   lin_vec(var_vec_2,21,0.5,2.5)
156   result_mat = new Matrix(var_vec_1.size(),var_vec_2.size())
157
158
159   //Begin multiple simulation for loop
160   for i=0, var_vec_1.size()-1 {
161     for j=0, var_vec_2.size()-1 {
162
163       //Reset all variables
164       reset_sim()
165
166       // Insert variable parameters here
167       acfreq = var_vec_1.x(i)
168       amplitude = var_vec_2.x(j)
169
170       //Init&Run simulation
171       init_sim()
172       if (graphical ==1) {
173       plot_vec(filevect_value,window_list,50,50,250,150)
174       plot_vec(acvect,window_list,50,450,250,150)
175       vectormovie(myelin_ref, axon_length, window_list, rangevarplot_list,550,50,250,150)
176       }
177       doNotify()
178       run()
179       close_all(window_manager)
180
181       //Simulation test functions here
182       block_test(apcount_timevec_list, result_mat, var_vec_1, var_vec_2, i, j, pulse_num,
            onset_response_time)
183
184
185
186       //End for loop here
```

```
187   }
188 }
189
190 //Plot Results
191 identify_success_intervals(result_mat,var_vec_2,var_vec_1,window_list,50,50,250,150)
192
193 //Write data
194   //Initialize Write
195   objref f2
196   f2 = new File()
197   f2.wopen("out.txt")
198   //Header
199   f2.printf("NEURON simulation data\n")
200   f2.printf("Model Parameters:\n")
201   f2.printf("Variable 1: from %f to %f\n",var_vec_1.x[0],var_vec_1.x[var_vec_1.size()-1])
202   f2.printf("Variable 2: from %f to %f\n",var_vec_2.x[0],var_vec_2.x[var_vec_2.size()-1])
203   f2.printf("Vector lengths:\n")
204   f2.printf("%f\t%f\n",var_vec_1.size(),var_vec_2.size())
205
206   //Data
207   for i=0, var_vec_1.size()-1 { for j=0, var_vec_2.size()-1 { f2.printf("%f\t%f\t%f\n",
          var_vec_1.x[i],var_vec_2.x[j],result_mat.x[i][j])  } }
208   //Close
209   f2.close()
```

`project_work_file_apspeeds`: this main file draws a plot of action potential conduction speed as a function of fiber diameter. Based on `project_work_file_block`.

```
1
2  //Adrien Rapeaux Imperial College Bioengineering 4th Year Individual Project
3  load_file("nrngui.hoc")
4  load_file("specify_sections.hoc")
5  load_file("assemble_axons.hoc")
6  load_file("stimvec_init.hoc")
7  load_file("implant_iclamps.hoc")
8  load_file("implant_apcounters.hoc")
9  load_file("acvectinit.hoc")
10 load_file("setblock.hoc")
11 load_file("comsolvector.hoc")
12 load_file("plot_vec.hoc")
13 load_file("vectormovie.hoc")
14 load_file("close_all.hoc")
15 load_file("plot_vec_vs.hoc")
16 load_file("log_vec.hoc")
17 load_file("lin_vec.hoc")
18 load_file("runcontrol.hoc")
19 load_file("ap_speed_measure.hoc")
20
21 //Integration Method: calling cvode.active() uses variable time step integration - Warning
       extremely slow
22 //cvode_active(1)
23 //Temperature
24 {celsius=20}
25
26 //Segmentation
27 node_segnum = 1   //Number of segments per node
28 myelin_segnum = 5  //Number of segments per half internode
29
30 //General Geometry Parameters
31 axon_diameter = 5 //(microns)diameter of small axon fibers in the frog ventral roots [1]
32 axon_length = 30000 //microns (this is actually a desired (MAX) length and not a set length
       due to the fixed length internodes)
33
34 diameter_internode_ratio = 100 //ratio of axon diameter (microns) to internode length (microns
       )
35 node_length = 3 //microns (this doesn't depend on axon diameter)
36
37 //Electrode positioning
38 electrode_pos = 0.5 //Position of electrode on axon as percent of total axon length
39 electrode_ypos = 1000 //microns distance from axon to electrode
40 strdef file_string
41 if (electrode_config == 2) {file_string = "efield_bipolar.txt"  //Electric field data for
       extracellular, bipolar electrode_version
42 } else {file_string = "efield_monopolar.txt"} //Electric field data for extracellular,
       monopolar electrode version
43 MIN_resolution = 1000   //We suppose that the electric field vector values are mapped at least
        every 1000 microns
44
45 //Neurophysiological Parameters
46 refractory_dur = 1.4    //[ms] duration of the refractory period (about 2ms in the frog)
47 onset_response_time = 20  //[ms] duration of the expected onset response to block (empirical
       value)
48
```

```
49  //Stimulation parameters
50  stim_delay = 5    //For measuring AP speeds, wait 5 ms for membrane voltage stabilization
        before sending out AP.
51  stim_amp = 10 //[mA]
52
53  pulse_delay = 5   //Delay between each stimulatory pulse [ms]
54  pulse_dur = 0.15  //Stimulation duration 0.15 ms
55  pulse_num = 1   //Number of action potentials to generate
56
57  //Simulation Parameters (Initialization)
58  graphical = 1 //This boolean determines whether the simulation uses graphics; useful for
        diagnostics
59  v_init = -70 //mV
60  timestep = 0.005  //ms
61  tinit = 0
62  tfinal = stim_delay+pulse_num*(pulse_dur+pulse_delay)+10
63
64  //AC stimulation (blocking) parameters
65  amplitude = 0  //Volts
66  acfreq = 4000 //Hz
67  acphase = 0   //rad
68  acdur = stim_delay+pulse_num*(pulse_dur+pulse_delay)  //ms make sure to block all APs to test
        block for an axon.
69  refractory_dur = 1.6  //ms duration of refractory period
70  acres = 1   //Resolution of acvector [percent] (1 = MAX)
71
72  //Capacitances
73  membrane_surfacic_capacitance = 2     // uF/cm
74  myelin_surfacic_capacitance = 0.005   // uF/cm
75
76  //Resistances
77  myelin_resistivity = 29         //MOhms.cm [2]
78  myelin_axial_resistance = myelin_resistivity/(PI*axon_diameter^2*((1/0.8)^2-1)*(10^-8))   //
        MOhms/cm calculated from resistivity and myelin cross-sectional area (assumed axon
        diameter to myelin diamter ratio: 0.75)
79  myelin_surfacic_conductance = 1/(10^5)  // mho/cm
80  axosplasmic_resistivity = 100     //Ohm.cm from standard set of parameters, need additional
        data to confirm for frog neurons
81  membrane_axial_resistance = 2e14    //Ohms/cm [2]
82
83  //Object Definitions (necessary at command level)
84  objref node_ref, myelin_ref, stimulator, stimvec, apcount, apcount_timevec_list, ap_speed_vect
        , fiber_diameter_vect, acvect, block_signal, veclist, filevect_position, filevect_value,
         vposition, vvalue, window_list, rangevarplot_list, window_manager, var_vec_1, var_vec_2
        , result_mat
85  create node[1], myelin[1]
86
87
88
89  //--------------------PROCEDURE SPACE----------------------
90
91
92
93  proc reset_sim() {
94  //Redefine objects to clear existing objects
95  objref node_ref, myelin_ref, stimulator, stimvec, apcount, apcount_timevec_list, ap_speed_vect
        , fiber_diameter_vect, acvect, block_signal, veclist, filevect_position, filevect_value,
         vposition, vvalue, window_list, rangevarplot_list, window_manager
96  create node[1], myelin[1]
97  access node[0]    //Define default section
98  window_list = new List()
99  rangevarplot_list = new List()
100 window_manager = new PWManager()
101
102 }
103
104 proc init_sim() {local i
105
106   //initialize simulation
107   myelin_length = axon_diameter*diameter_internode_ratio/2  //Myelin sections are defined in
          half-internodes
108   nb_node = int(axon_length/(node_length+axon_diameter*diameter_internode_ratio))
109   print "the axon contains ", nb_node, "nodes"
110   total_length = nb_node*(node_length+2*myelin_length)
111   print "the total length of the axon is ", total_length, "microns"
112   create node[nb_node], myelin[2*nb_node]
113   // SectionReference lists must be made at command level since sections cannot be
114   //directly passed as variables
115   node_ref = new List()
116   myelin_ref = new List()
117   //Fill SectionReference Lists
118   for i=0, nb_node-1 {
119     node[i] node_ref.append(new SectionRef())
120     myelin[2*i] myelin_ref.append(new SectionRef())
```

```
121       myelin[2*i+1] myelin_ref.append(new SectionRef())
122       }
123
124     specify_sections(node_ref, myelin_ref, node_segnum, myelin_segnum, axon_diameter,
            node_length, myelin_length, membrane_surfacic_capacitance, myelin_surfacic_capacitance,
            axosplasmic_resistivity, myelin_axial_resistance, membrane_axial_resistance)
125     assemble_axons(nb_node, node_ref, myelin_ref)
126     stimvec_init(stim_delay, timestep, pulse_delay, pulse_dur, pulse_num, stimvec)
127     implant_iclamps(node_ref, 0, stimulator, stimvec, stim_amp, axon_diameter, timestep, tfinal)
128     implant_APcounters(node_ref, nb_node, apcount, apcount_timevec_list)
129     COMSOLvector(filevect_value, filevect_position, file_string, electrode_pos, total_length,
            MIN_resolution, vposition, vvalue, electrode_ypos, node_length, myelin_length)
130     adjusted_block_time = acvectinit(acdur, acres, timestep, acfreq, acphase, amplitude, acvect,
             block_signal)
131     setblock(veclist, myelin_ref, block_signal, myelin_length, MIN_resolution, vposition, vvalue
            , node_ref, nb_node, node_length, timestep, acres)
132
133     init()
134     if (graphical == 1) {
135       runcontrol(v_init, tinit, tfinal, timestep, 1197, 262)
136     } else {
137       v_init = -70 //mV
138       tstop = tfinal
139       dt = timestep
140       t = tinit
141     }
142   }
143
144
145
146   // ------------------------COMMAND SPACE-------------------
147
148
149   //Preliminary Commands
150   amplitude = 0 //Turn off blocking for this simulation
151
152
153   //Define variable vectors containing increments (might need function in future)
154
155   lin_vec(var_vec_1,181,0.1,2)
156   result_mat = new Matrix(1,var_vec_1.size())
157
158
159   //Begin multiple simulation for loop
160   for i=0, var_vec_1.size()-1 {
161
162       //Reset all variables
163       reset_sim()
164
165       // Insert variable parameters here
166       axon_diameter = var_vec_1.x(i)
167
168
169       //Init&Run simulation
170       init_sim()
171       if (graphical ==1) {
172       plot_vec(filevect_value,window_list,50,50,250,150)
173       plot_vec(acvect,window_list,50,450,250,150)
174       vectormovie(myelin_ref, axon_length, window_list, rangevarplot_list,550,50,250,150)
175       }
176       doNotify()
177       run()
178       close_all(window_manager)
179
180       //Simulation test functions here
181       result_mat.x[0][i] = ap_speed_measure(apcount_timevec_list,0,2,0,int(nb_node*electrode_pos
            ),node_length+diameter_internode_ratio*axon_diameter)
182       print "At fiber diameter: ", var_vec_1.x[i], "measured AP speed was: ", result_mat.x[0][i]
183
184
185
186       //End for loop here
187   }
188
189   //Plot Results
190     objref result_vec
191     result_vec = new Vector()
192     result_mat.getrow(0,result_vec)
193     plot_vec_vs(result_vec,var_vec_1,window_list,50,50,250,150)
194
195   //Write data
196     //Initialize Write
197     objref f2
198     f2 = new File()
```

```
199     f2.wopen("ap_speeds_04_06.txt")
200     //Header
201     f2.printf("NEURON simulation data\n")
202     f2.printf("Model Parameters:\n")
203     f2.printf("Variable 1: from %f to %f\n",var_vec_1.x[0],var_vec_1.x[var_vec_1.size()-1])
204     f2.printf("Variable 2: Not Applicable\n")
205     f2.printf("Vector lengths: %f\n",var_vec_1.size())
206
207     //Data
208     for i=0, var_vec_1.size()-1 {
209
210         f2.printf("%f\t%f\n",var_vec_1.x[i],result_mat.x[0][i])
211
212     }
213     //Close
214     f2.close()
```

`nrngui.hoc`:this file is included within NEURON.

`specify_sections`

```
 1
 2  //Specify section properties
 3  //Uses model parameters, hardcoded parameters (note ion concentrations)
 4  //and computed parameters from geometry_comput to fill in all necessary
 5  //section properties for nodes of Ranvier and internodes
 6  //myelin-covered internodes are considered perfect insulators (no passive channels)
 7  //Only nodes are given Frankenhaeuser-Huxley channels
 8  //All other parameters are specified for xenopus laevis myelinated axons
 9  /*Arguments:
10  node_ref = $o1
11  myelin_ref = $o2
12  node_segnum = $3
13  myelin_segnum = $4
14  axon_diameter_small = $5
15  node_length = $6
16  myelin_length_small = $7
17  membrane_surfacic_capacitance = $8
18  myelin_surfacic_capacitance = $9
19  axosplasmic_resistivity = $10
20  myelin_axial_resistance_small = $11
21  membrane_axial_resistance = $12
22
23  */
24
25  proc specify_sections() {local i
26
27    for i=0,$o1.count()-1 {
28    $o1.o(i).sec {
29      L = $6
30      nseg = $3
31      Ra = $10
32      diam = $5
33      cm = $8
34      insert fh
35      nai=13.74
36      nao=114.5
37      ki=120
38      ko=2.5
39      insert extracellular
40      xg = 1e10
41      xc = 0    //There is no myelin over nodes, hence no resistance nor capacitance effects
42      xraxial = 1e9
43      e_extracellular = 0
44      }
45    }
46
47    for i=0,$o2.count()-1 {
48    $o2.o(i).sec {
49      L = $7
50      nseg = $4
51      Ra = $10
52      diam = $5
53      cm = $9
54      insert extracellular
55      xraxial = 1e9
56      xc = 0    //Membrane capacitance was set to reflect myelin
57      xg = 0    //Myelin is treated as a perfect insulator
58      e_extracellular = 0
59      }
60    }
61  }
```

### assemble_axons

```
1
2  //Axon assembly procedure:
3  //We have individual sections: nodes and then half-internodes
4  //(in order to avoid having a node on the extremity of the model)
5  //We assemble units by connecting two half-internodes together with
6  //a node between them (myelin_node_myelin)
7  //We then assemble these units together into the finished axon (myelin_myelin)
8  //
9  /*
10 Arguments:
11 nb_node_small = $1
12 node_small_ref = $o2
13 myelin_small_ref = $o3
14 */
15
16 proc assemble_axons() {local i
17
18   //Assembly of small axon: myelin-node-myelin
19   for i=0, $1-1 {
20   $o3.o(2*i).sec connect $o2.o(i).sec(0), 1
21   $o2.o(i).sec connect $o3.o(2*i+1).sec(0), 1
22   }
23
24   //Assembly of small axon: myelin-myelin
25   for i=0, $1-2 {
26   $o3.o(2*i+1).sec connect $o3.o(2*i+2).sec(0), 1
27   }
28 }
```

### stimvec_init

```
1
2  //STIMVEC_INIT
3  //Initializes the vector of pulses for the
4  //stimulation electrode. When used with the
5  //vector.play method, the vector created
6  //by this procedure allows a train of action
7  //potentials to be created by the stimulation
8  //electrode.
9
10 /*Arguments
11 $1 = initial delay before starting the pulse train  [ms]
12 $2 = timestep of the simulation           [ms]
13 $3 = delay between two pulses           [ms]
14 $4 = duration of a pulse              [ms]
15 $5 = number of pulses
16 $o6 = object reference for vector
17 */
18
19 proc stimvec_init() {local dur, i, j
20
21
22   dur = $1+$5*($3+$4)
23
24   $o6 = new Vector(int(dur/$2),0)     //vector creation with one value for every timestep in
       its duration
25   for i=0,$5-1  {
26     for j=0,int($4/$2)-1  {
27       $o6.x[$1/$2+i*$3/$2+j] = 1  //value of 1 is initial then adjusted in implant_iclamps
28     }
29   }
30
31 }
```

### implant_iclamps

```
1
2  //IMPLANT ICLAMPS
3  //Procedure to set IClamp point processes at
4  //the correct locations in modeled axons.
5  //Serves to modelise intracellular electrodes.
6  //Uses stimulation vectors (initialized in
7  //stimvec_init.hoc) to drive the amplitude of
8  //the current clamps (IClamps)
9
10 /*Arguments:
11 node_ref = $o1
12 node_number = $2
13 stim = $o3
14 stimvec = $o4
15 stim_amp = $5
```

```
16  axon_diameter = $6
17  timestep = $7
18  tfinal = $8
19  */
20
21  proc implant_iclamps() {
22
23  $o1.o($2).sec $o3 = new IClamp(.5)  //Stimulator electrode connected at halfpoint of first
        node
24  $o1.o($2).sec $o3.dur = $8
25  $o1.o($2).sec $o4.mul($5*$6/5).play(&$o3.amp,$7)
26
27  }
```

### implant_apcounters

```
1
2   //Implant APcounters
3   //Procedure responsible for inserting the APcounter
4   //point mechanisms at the specified places. Uses lists for
5   //flexibility. Implants 3 APCounters per axon, one at each
6   //extremity of the axon and one at its middle.
7   //Arguments
8   //node_small_ref = $o1
9
10  //nb_node_small = $2
11
12  //apcount_small = $o3
13
14  //apcount_timevec_list_small = $o4
15
16
17  proc implant_APcounters() {local i localobj timevec
18
19  $o3 = new List()
20
21  $o4 = new List()
22
23
24  $o1.o(0).sec $o3.append(new APCount(0.5))
25  $o1.o($2-1).sec $o3.append(new APCount(0.5))
26  $o1.o(int($2/2)).sec $o3.append(new APCount(0.5))
27
28
29  for i=0, $o3.count()-1 {
30    $o3.o(i).thresh = 32  //mV
31    timevec = new Vector()
32    $o4.append(timevec)
33    $o3.o(i).record($o4.o(i))
34  }
35  //Threshold set arbitrarily: needs to avoid misfire but be sure to count all APs
36  //ignoring differences in amplitude (should be same for all APs regardless)
37
38  }
```

### acvectinit

```
1
2   //ACVECTINIT
3   //Initialization procedure for ac blocking signal vectors (for vector.play)
4   //This procedure creates a vector that will drive the
5   //external electric field using a sine wave. With
6   //timestep (dt), the vector is created with the
7   //length and frequency specified by acfreq and acdur,
8   //with phase adjustment to make sure the last oscillation
9   //does not cause unwanted stimulation at block cutoff
10  //due to the field value remaining at the last
11  //value specified by the vector. In this case it
12  //is 0, ensuring correct cutoff of the sine signal.
13
14  /*Arguments:
15  acdur = $1
16  acres = $2
17  timestep = $3
18  acfreq = $4
19  acphase = $5
20  amplitude = $6
21  acvect = $o7
22  ac_small = $o8
23  */
24
25  func acvectinit() {local dur, oscillation_time, diff, phase_adjust
26
```

```
27    oscillation_time = 1/($4)*1000  // [ms]
28    dur = int($1/oscillation_time)*oscillation_time //calculation of the time for the closest
         number of full oscillations (rounded down)
29    print "adjusted block time is: ",dur
30    $o7 = new Vector(int(dur*$2/$3),0)    //initial vector initialization
31    $o7.sin($4,$5,$3/$2)      //assignment of sinus values (freq,phase,dt) to vector
32    $o7.x[$o7.size()-1] = 0             //set extracellular field to 0 when electrode
         disconnected
33    $o8=$o7.c.mul($6)             //Amplitude factor to adjust voltage, supposing system is
         linear
34    return dur
35  }
```

### setblock

```
1
2   //SETBLOCK
3   //Assignment of extracellular electric field values for blocking signal
4   /*Arguments
5   veclist_small = $o1
6   myelin_small_ref = $o2
7   ac_small = $o3
8   myelin_length_small = $4
9   MIN_resolution = $5
10  vposition_small = $o6
11  vvalue_small = $o7
12  node_small_ref = $o8
13  nb_node_small = $9
14  node_length = $10
15  timestep = $11
16  acres = $12
17  */
18  load_file("find_segment_coordinates_singleaxon.hoc")
19  load_file("find_vect_coord.hoc")
20
21
22  proc setblock() {local x, segment_coordinate, Efieldvect_index, list_index, i localobj tmpvec
23
24
25
26    $o1 = new List()
27
28    for i=0, $9-1 {
29      $o2.o(2*i).sec {
30        for (x,1) {
31        tmpvec = $o3.c
32        segment_coordinate = find_segment_coordinates(0,i,x,$4,$10)
33        Efieldvect_index = find_vect_coord(segment_coordinate,$5,$o6)
34        tmpvec.mul($o7.x[Efieldvect_index])
35        list_index=$o1.append(tmpvec)
36        $o1.o(list_index-1).play(&e_extracellular(x),$11/$12)
37
38        }
39      }
40
41      $o8.o(i).sec {
42        for (x,1) {
43        tmpvec = $o3.c
44        segment_coordinate = find_segment_coordinates(1,i,x,$4,$10)
45        Efieldvect_index = find_vect_coord(segment_coordinate,$5,$o6)
46        tmpvec.mul($o7.x[Efieldvect_index])
47        list_index=$o1.append(tmpvec)
48        $o1.o(list_index-1).play(&e_extracellular(x),$11/$12)
49        }
50      }
51
52      $o2.o(2*i+1).sec {
53      for (x,1) {
54        tmpvec = $o3.c
55        segment_coordinate = find_segment_coordinates(2,i,x,$4,$10)
56        Efieldvect_index = find_vect_coord(segment_coordinate,$5,$o6)
57        tmpvec.mul($o7.x[Efieldvect_index])
58        list_index=$o1.append(tmpvec)
59        $o1.o(list_index-1).play(&e_extracellular(x),$11/$12)
60        }
61      }
62    }
63  }
```

### comsolvector

```
1
2   //COMSOLvector
```

```
3  //Procedure for loading and treatment of COMSOL data; yields modified vposition and vvalue,
       one for each axon
4  /*Arguments
5  filevect_value = $o1
6  filevect_position = $o2
7  file_string = $s3
8  electrode_pos = $4
9  total_length = $5
10 MIN_resolution = $6
11 vposition = $o7
12 vvalue = $o8
13 electrode_ypos = $9
14 node_length = $10
15 myelin_length = $11
16 */
17
18 load_file("find_vect_coord.hoc")
19 load_file("comsolfile_read.hoc")
20
21 proc COMSOLvector() {local i
22
23 //Retrieval of Electric Field Simulation data (file needs to contain only column vectors (
       space-separated values from vector to vector)
24 elmt_num = comsolfile_read($o2,$o1,$s3,$9)
25
26 //Determine landmarks for filevect
27 //Determine start and end points
28 //We supposed middle point is at point of strongest Efield
29 middle_point_index = $o1.max_ind()
30 middle_point = $o2.x[middle_point_index]
31 start = middle_point-$4*$5
32 end = middle_point-($4-1)*$5
33
34
35 //Find start and end locations on the filevect in order to truncate it to the correct size for
        the axons
36 vec_start_index = find_vect_coord(start,$6,$o2)
37 vec_end_index = find_vect_coord(end,$6,$o2)
38 vec_total_length = abs($o2.x[vec_start_index]-$o2.x[vec_end_index])
39
40 spatial_length = ($o2.x[$o2.size-1]-$o2.x[0])
41
42 //Length of the truncated filevect to fit for the model's axon's length
43 total_index_length = abs(vec_end_index-vec_start_index)
44
45 //Create new vectors for the subset of COMSOL's data that will be used in the simulation
46 $o7 = new Vector(total_index_length,0)
47 $o8 = new Vector(total_index_length,0)
48
49 //Truncate Electric Field data to fit for specified axon length
50 $o7.copy($o2,vec_start_index,vec_end_index)
51 $o8.copy($o1,vec_start_index,vec_end_index)
52
53 //Adjust vposition to reflect model coordinates (starting at 0)
54 vposition_start_pos = $o7.x[0]
55 for i=0,$o7.size()-1 {
56   $o7.x[i]=$o7.x[i]-vposition_start_pos
57 }
58
59 //Additional Equivalent index computations
60 node_index_length = int(total_index_length*node_length/$5)
61 myelin_index_length = int(total_index_length*$11/$5)
62
63 //Debugging printouts
64 print "total_index_length is ", total_index_length
65 print "node_index_length is ", node_index_length
66 print "myelin_index_length is ", myelin_index_length, "values for an equivalent ", $11, "
       microns"
67 print "vvalue has ", $o8.size, "values for an equivalent ", vec_total_length, "microns"
68
69 }
```

### find_vect_coord

```
1
2  //function for finding the closest location on the filevect for a certain coordinate:
3  //Bruteforce search for the minimum distance between specified segment coordinates and
       filevect_pos coordinates
4  //Returns filevect index, used to access filevect_value at the correct location
5
6  //Arguments:
7  //$1 is the coordinate in microns which we are looking for in the position vector
8  //$2 is a "high number" compared to the resolution in microns of the vector
```

```
9  //so that is can be used adequately for finding the minimum distance between the desired
       coordinate and coordinates on the position vector
10 //$o3 is the position vector
11
12 //The function returns the index of the position vector that contains the closest value to
       desired_coordinate
13
14 func find_vect_coord() {local desired_coordinate,MIN_resolution,vect_index,i
15   desired_coordinate = $1
16   MIN_resolution = $2
17   vect_index = 0    //Always returns something even if 0
18   for i=0, $o3.size()-1 {
19   if (abs(desired_coordinate-$o3.x[i])<MIN_resolution) {
20     MIN_resolution = abs(desired_coordinate-$o3.x[i])
21     vect_index = i
22     }
23
24   }
25   return vect_index
26 }
```

### comsolfile_read

```
1
2  //COMSOLFILE_READ
3  //Description: reads a COMSOL-generated vector file
4  //Vector file syntax is two columns with column 1 as position vector and column 2 as value
       vector (remove COMSOL headers)
5  //COMSOL units are supposed to be centimetres:
6  //The corresponding correction factor has been specified to convert cm into microns for NEURON
7  //It is supposed that the values vector is already in V/m, bypassing the need for conversion
       for NEURON
8
9  //Arguments:
10 //$o1 is the position vector argument (initialized as new Vector())
11 //$o2 is the value vector argument
12 //$s3 is the name of the file to read as a string
13 //$4 is the target y (vertical position) of the vector we want from the COMSOL matrix
14 //The function returns the size of both vectors (both have the same size)
15
16 //Function
17 func comsolfile_read() {local conversion_factor, elmt_num, target_y, resolution, xval, yval,
       value localobj f1
18
19   //Conversion Factor
20   conversion_factor = 10000 //Convert centimetres to microns
21   resolution = 0.0025 //Half the step used in COMSOL for meshing (cm) (is 50 microns per step)
22   target_y = $4/conversion_factor //transfer to COMSOL coordinates (cm)
23   f1 = new File()
24   f1.ropen($s3)
25   strdef s1
26   xval=0
27   yval=0
28   value=0
29
30   $o1 = new Vector()
31   $o2 = new Vector()
32   while (f1.gets(s1) >= 0) {
33     sscanf(s1,"%lf%lf%lf", &yval, &xval, &value)
34     if (yval>(1-target_y-resolution) && yval<(1-target_y+resolution)) {
35       $o1.append(xval)
36       $o2.append(value)
37     }
38   }
39   //Convert COMSOL cm to NEURON microns in position vector
40   $o1.mul(conversion_factor)
41
42   return $o2.size()
43 }
```

### plot_vec

```
1
2  //PLOT_VEC
3  //Plots specified vector and appends in specified object list
4  //for reference
5  //Arguments:
6  //$o1 = vector to be plotted
7  //$o2 = object (window) list
8  //$3 = window x position
9  //$4 = window y position
10 //$5 = window width
11 //$6 = window height
```

```

proc plot_vec() {local minval, maxval localobj window

minval = $o1.min()
maxval = $o1.max()
window = new Graph(0)
window.view(0,minval,$o1.size()-1,maxval,$3,$4,$5,$6)
window.size(0,$o1.size()-1,minval,maxval)
$o1.plot(window)
$o2.append(window)
}
```

### vectormovie

```

//VECTORMOVIE
//Makes an updating shapeplot of an axon's membrane
//potential for visualization of action potentials
//and blocks during the simulation

//Arguments:
//$o1 = sectionref list pointing to the axon of interest (myelin)
//$2 = desired total length of the axon (microns) (real length is always smaller)
//$o3 = object (window) list
//$o4 = object (RangeVarPlot) list
//$5 = window x position
//$6 = window y position
//$7 = window width
//$8 = window height

proc vectormovie() {local extremeval localobj window, plot

//We arbitrarily specify 150 mV as the max membrane potential
//during action potentials and blocking, so as to keep the
//same scale when comparing multiple axons during simulation.
extremeval = 150




window = new Graph(0)
window.view(0,-extremeval,$2,extremeval,$5,$6,$7,$8)
window.size(0,$2,-extremeval,extremeval)
window.save_name("flush_list.")
flush_list.append(window)
$o3.append(window)
plot = new RangeVarPlot("v")
$o1.o(0).sec plot.begin(1)
$o1.o($o1.count()-1).sec plot.end(1)
plot.origin(0)
window.addobject(plot, 2, 1, 0.7, 0.9)
$o4.append(plot)

}
```

### close_all

```

//CLOSE_ALL
//Closes all windows to refresh the GUI in between simulations
//NOTE: closes neuron gui as well, leaving only the terminal

//Arguments
//$o1 = window manager object

proc close_all() {
 while ($o1.count()>2) {
 $o1.close(1)
 }
}
```

### log_vec

```

//LOG_VEC
//Procedure to modify objref into
//vector with logarithmically increasing values,
//for simulation and plotting purposes

//Arguments:
//$o1 : objref to be used
//$2: number of decades
```

```
//$3: number of values per decade (1 to 4)
//$4: starting value (advised to be power of 10)
//    NOTE: values are 1,2,5,7
proc log_vec() {local i, starting_decade
  $o1 = new Vector($2*$3)
  for i=0, $2-1 {
    $o1.x[i*$3] = 1*$4*10^(i)
    if ($3>1) $o1.x[i*$3+1] = 2*$4*10^(i)
    if ($3>2) $o1.x[i*$3+2] = 5*$4*10^(i)
    if ($3>3) $o1.x[i*$3+3] = 7*$4*10^(i)
    ///UNFINISHED
  }
}
```

### lin_vec

```

//LIN_VEC
//Procedure to modify objref into vector with linearly
//increasing values for simulation and plotting

//Arguments:
//$o1 : objref to be used
//$2 : number of elements in vector
//$3 : value step
//$4 : starting value

proc lin_vec() {local i
  $o1 = new Vector($2)
  for i=0, $2-1 {
    $o1.x[i] = $4+i*$3
  }
}
```

### plot_vec_vs

```

//PLOT_VEC_VS
//Procedure to plot a vector versus another vector
//Arguments:
//$o1 = vector to be plotted (y axis)
//$o2 = vector to be plotted (x axis)
//$o3 = object (window) list
//$4 = window x position
//$5 = window y position
//$6 = window width
//$7 = window height

proc plot_vec_vs() {local minval, maxval localobj window

minval = $o1.min()
maxval = $o1.max()
window = new Graph(0)
window.view(0,minval,$o1.size()-1,maxval,$4,$5,$6,$7)
window.size($o2.x(0),$o2.x($o2.size()-1),minval,maxval)
$o1.plot(window, $o2)
$o3.append(window)
}
```

### identify_success_intervals

```

//IDENTIFY_SUCCESS_INTERVALS
//Procedure working with the results matrix and the two
//variable vectors to yield a list of y_vectors and x_vectors
//to be plotted (generally in the same window). Vectors are
//checked in size to be bigger than one element (dot plots
//are invisible) and contiguous. X values for which there is
//no success (nothing to be plotted) are skipped and
//new vectors are appended to the list of plotted vectors
//in order for there to be no line on the plot for these
//values (due to NEURON plotting non-contiguous vectors
//with this not being visible to the user)
//are individually appended to a List for plotting,
//effectively ignoring and not plotting for variable ranges
//in which there is no success. Plotted vectors are on
//single window, scaled to contain everything.

//Arguments
//$o1 : the results matrix with rows for x and columns for y
//$o2 : the y axis vector
//$o3 : the x axis vector
```

```
22  //$o4 : object (window) list
23  //$5 : window x position
24  //$6 : window y position
25  //$7 : window width
26  //$8 : window height
27
28  proc identify_success_intervals() {local i,j, minx, miny, maxx, maxy localobj rowvector,
        index_vec, tmpvec3, interval_xvector, interval_start_yvector, interval_end_yvector,
        window
29    rowvector = new Vector()
30    index_vec = new Vector()
31    tmpvec3 = new Vector()
32    interval_xvector = new List()
33    interval_start_yvector = new List()
34    interval_end_yvector = new List()
35    interval_xvector.append(tmpvec3.c)
36    interval_start_yvector.append(tmpvec3.c)
37    interval_end_yvector.append(tmpvec3.c)
38    for i=0, $o1.nrow()-1 {
39      rowvector = $o1.getrow(i)
40
41      rowvector.printf()
42      if (rowvector.contains(1)) {
43        index_vec = rowvector.indvwhere("==",1)
44        if (index_vec.size()>1) {
45          if (index_vec.x[index_vec.size()-1]-index_vec.x[0]==index_vec.size()-1) {
46
47            interval_xvector.o(interval_xvector.count()-1).append($o3.x[i])
48
49            interval_start_yvector.o(interval_xvector.count()-1).append($o2.x[index_vec.x[0]])
50
51            interval_end_yvector.o(interval_xvector.count()-1).append($o2.x[index_vec.x[
        index_vec.size()-1]])
52
53          }
54        }
55
56      } else {
57        interval_xvector.append(tmpvec3.c)
58        interval_start_yvector.append(tmpvec3.c)
59        interval_end_yvector.append(tmpvec3.c)
60      }
61    }
62    minx = $o3.x[0]
63    maxx = $o3.x[$o3.size()-1]
64    miny = $o2.x[0]
65    maxy = $o2.x[$o2.size()-1]
66    window = new Graph(0)
67    window.view(minx,miny,maxx,maxy,$5,$6,$7,$8)
68    window.size(minx,maxx,miny,maxy)
69    for i=0,interval_xvector.count()-1 {
70      interval_start_yvector.o(i).line(window,interval_xvector.o(i),3,1)
71      interval_end_yvector.o(i).line(window,interval_xvector.o(i),2,1)
72    }
73    $o4.append(window)
74  }
```

## read_action_potential_speeds

```
1
2   //READ_ACTION_POTENTIAL_SPEEDS
3   //Description: reads a file containing action potential
4   //propagation speeds as function of fiber diameter
5   //diameters are in microns and speeds are in m/s
6   //NEURON units are ms and microns for time and space
7   //correction factors for the speeds (x1000) have been
8   //implemented in this procedure.
9
10  //Arguments:
11  //$s1 name of file containing AP propagation speeds as function of fiber diameter
12  //$o2 is the speed vector argument (objref)
13  //$o3 is the fiber diameter vector argument (objref)
14
15
16  //Function
17  proc read_action_potential_speeds() {local i, conversion_factor, diameter_val, speed_val
        localobj f1
18
19    //Conversion Factor
20    conversion_factor = 1000
21    f1 = new File()
22    f1.ropen($s1)
23    strdef str
24    speed_val=0
```

```
25    diameter_val=0
26
27    $o2 = new Vector()
28    $o3 = new Vector()
29
30    for i=1, 5 {
31      f1.gets(str) //Skip file header
32    }
33
34    while (f1.gets(str) >= 0) {
35      sscanf(str,"%lf%lf", &diameter_val, &speed_val)
36      $o2.append(speed_val)
37      $o3.append(diameter_val)
38
39    }
40
41    $o2.mul(conversion_factor)
42
43  }
```

### determine_stim_timing

```
1
2   //DETERMINE_STIM_TIMING
3   //Procedure to determine the time at which the
4   //Action Potentials for selective stimulation
5   //should be generated, based on the projected
6   //cutoff time for the blocking signal. Since the
7   //blocking signal needs to be charge balanced
8   //and that the selective stimulation protocol
9   //requires that the block reach a steady state,
10  //leading to a waiting period, based on the
11  //desired blocking times an adjusted block time
12  //is determined by acvectinit.hoc . This
13  //procedure uses this adjusted block time to
14  //time the generation of the APs appropriately
15  //so that the faster AP will be annihilated by
16  //the block and the slower AP will make it through.
17  //The function takes into account the propagation
18  //speeds of the action potentials read off of a
19  //file, as well as a set refractory period. It
20  //calculates a stimulation timing based on the mean
21  //of the times at which the faster and slower
22  //AP potentials reach the blocking site respectively,
23  //and substracts the refractory period. It then
24  //substracts this value to the projected block time
25  //in order to obtain the time at which the APs need
26  //to be generated.
27  //Usage is thus acdur = determine_block_timing(parameters).
28
29  //Arguments:
30  //$o1 = ap speeds vector [microns/ms]
31  //$o2 = fiber diameter vector (for ap speeds) [microns]
32  //$3 = "small" axon fiber diameter
33  //$4 = "large" axon fiber diameter
34  //$5 = total small axon length
35  //$6 = blocking electrode position for small axon
36  //$7 = total large axon length
37  //$8 = blocking electrode position for large axon
38  //$9 = total small internode length (node length + internode length)
39  //$10 = total large internode length (node length + internode length)
40  //$11 = duration of refractory period (ms)
41  //$12 = adjusted block cutoff time (ms)
42
43
44  func determine_stim_timing() {local apspeed_small, apspeed_large, dist_small, dist_large,
        time_arrival_slow, time_arrival_fast, timing
45    apspeed_small = $o1.x[$o2.indwhere("==",$3)]
46    apspeed_large = $o1.x[$o2.indwhere("==",$4)]
47    dist_small = $5*$6-$9/2 //microns
48    dist_large = $7*$8-$10/2 //microns
49    time_arrival_slow = dist_small/apspeed_small
50    print "ETA slow AP: ", time_arrival_slow
51    time_arrival_fast = dist_large/apspeed_large
52    print "ETA fast AP: ", time_arrival_fast
53    timing = $12-((time_arrival_slow+time_arrival_fast)/2-$11)
54    print "timing chosen is :",timing
55    return timing
56
57  }
```

### runcontrol

```
1
2  //RUNCONTROL
3  //Procedure to create the runcontrol window with
4  //parameters to avoid changing dt, while providing
5  //useful tools for tweaking the simulation on the
6  //fly.
7
8  //Arguments
9  //v_init = $1
10 //t = $2
11 //tstop = $3
12 //dt = $4
13 //panel_x_position = $5
14 //panel_y_position = $6
15
16
17
18 proc runcontrol() {
19
20 xpanel("RunControl", 0)
21 v_init = $1
22 xvalue("Init","v_init", 1,"stdinit()", 1, 1 )
23 xbutton("Init & Run","run()")
24 xbutton("Stop","stoprun=1")
25 runStopAt = 5
26 xvalue("Continue til","runStopAt", 1,"{continuerun(runStopAt) stoprun=1}", 1, 1 )
27 runStopIn = 1
28 xvalue("Continue for","runStopIn", 1,"{continuerun(t + runStopIn) stoprun=1}", 1, 1 )
29 xbutton("Single Step","steprun()")
30 t = $2
31 xvalue("t","t", 2 )
32 tstop = $3
33 xvalue("Tstop","tstop", 1,"tstop_changed()", 0, 1 )
34 dt = $4
35 xvalue("dt","dt", 1,"setdt()", 0, 1 )
36 steps_per_ms = int(1/$4)
37 xvalue("Points plotted/ms","steps_per_ms", 1,"setdt()", 0, 1 )
38 screen_update_invl = 0.05
39 xvalue("Scrn update invl","screen_update_invl", 1,"", 0, 1 )
40 realtime = 0
41 xvalue("Real Time","realtime", 0,"", 0, 1 )
42 xpanel($5,$6)
43
44 }
```

### success_test

```
1
2  //SUCCESS_TEST
3  //Function returns boolean using a test to determine
4  //if selective stimulation really occurred during the
5  //simulation. It aims at filtering out cases where:
6  //1. Block was partial/inexistent, leading to generation
7  //of additional APs at either the small or large axons
8  //2. Stimulated AP didn't make it through small axon
9  //3. Stimulated AP made it through large axon
10
11 //If none of the above are true then the test returns 1 (true)
12 //else it returns false.
13
14 //APCounter time vectors contain the times in the simulation
15 //at which the counters were triggered. By using logic it
16 //is possible to test for cases where selective stimulation
17 //was achieved, categorizing any other case as a failure.
18 //The test ignores any action potentials from the block
19 //onset response by creating another vector containing
20 //the time stamps for action potential detection and
21 //removing any that were detected before onset_response_time.
22 //For the small axon, both APCounters should be triggered
23 //1 time, corresponding to an AP travelling distally
24 //from the proximal end, hence the trigger time for the
25 //proximal APCounter should be earlier than for the distal
26 //APCounter. For the large axon the proximal APCounter
27 //should be triggered only one time, and never the distal one.
28 //Delay is computed using geometry parameters from command
29 //level as the procedure is called by the main program.
30 //An expected delay between ap timestamps for the small
31 //axon is calculated and compared to the delay between the
32 //two timestamps in the APCounter time vectors.
33 //If it is within a 1 ms margin, the test is deemed successful.
34
35
36 //Arguments
37 //$o1 = list of apcounter time vectors for small axon
```

```
38  //$o2 = list of apcounter time vectors for large axon
39  //$o3 = matrix of results
40  //$o4 = vector for variable 1 values (x values)
41  //$o5 = vector for variable 2 values (y values)
42  //$6 = value of variable 1 at which the test is made (index of current x in x value vector)
43  //$7 = value of variable 2 at which the test is made (index of current y in y value vector)
44  //$o8 = ap speeds vector //[microns/ms]
45  //$o9 = fiber diameter vector (for ap speeds) [microns]
46  //$10 = "small" axon fiber diameter
47  //$11 = proximal APcounter node number (small axon)
48  //$12 = distal APcounter node number (small axon)
49  //$13 = total small internode length (node length + internode length)
50  //$14 = onset response delay in [ms]
51
52  proc success_test() {local small_axon_APpropagation_speed, small_APCounter_distance,
        expected_delay, i localobj apcount_timevec_proximal_small, apcount_timevec_distal_small,
         apcount_timevec_proximal_large, apcount_timevec_distal_large
53
54    //Remove any ignored values within the vectors recording detected APs
55    apcount_timevec_proximal_small = new Vector()
56    apcount_timevec_distal_small = new Vector()
57    apcount_timevec_proximal_large = new Vector()
58    apcount_timevec_distal_large = new Vector()
59
60    if ($o1.o(0).size()>=1 && $o1.o(1).size()>=1) {
61      for i=0,$o1.o(0).size()-1 {
62        if ($o1.o(0).x[i] > $14) {
63          apcount_timevec_proximal_small.append($o1.o(0).x[i])
64        }
65
66      }
67      for i=0,$o1.o(1).size()-1 {
68        if ($o1.o(1).x[i] > $14) {
69          apcount_timevec_distal_small.append($o1.o(1).x[i])
70        }
71      }
72      for i=0,$o2.o(0).size()-1 {
73        if ($o2.o(0).x[i] > $14) {
74          apcount_timevec_proximal_large.append($o2.o(0).x[i])
75        }
76
77      }
78      for i=0,$o2.o(1).size()-1 {
79        if ($o2.o(1).x[i] > $14) {
80          apcount_timevec_distal_large.append($o2.o(1).x[i])
81        }
82      }
83
84
85      if (apcount_timevec_proximal_small.size()==1 && apcount_timevec_distal_small.size()==1 &&
         apcount_timevec_proximal_large.size()==1 && apcount_timevec_distal_large.size()==0) {
86        small_axon_APpropagation_speed = $o8.x[$o9.indwhere("==",$10)]
87        small_APCounter_distance = ($12-$11)*$13
88        expected_delay = small_APCounter_distance/small_axon_APpropagation_speed
89        if (apcount_timevec_distal_small.x(0)-apcount_timevec_proximal_small.x(0)>expected_delay
         -1 && apcount_timevec_distal_small.x(0)-apcount_timevec_proximal_small.x(0)<
         expected_delay+1) {
            $o3.x[$6][$7]=1
91          print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed
         selective stim was successful."
92      } else {
93          $o3.x[$6][$7]=0
94          print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed
         selective stim failed.\nReason: timing mismatch\nExpected delay was: ", expected_delay,
         "while real delay was: ", apcount_timevec_distal_small.x(0)-
         apcount_timevec_proximal_small.x(0)
95      }
96
97      } else {
98        $o3.x[$6][$7]=0
99        print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed selective
         stim failed.\nReason: AP count mismatch.\nCode: ",apcount_timevec_proximal_small.size(),
         apcount_timevec_distal_small.size(),apcount_timevec_proximal_large.size(),
         apcount_timevec_distal_large.size()
100     }
101   } else {
102     $o3.x[$6][$7]=0
103     print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed selective
         stim failed.\nReason: APs annihilated/missing APs"
104   }
105 }
```

### success_test_bipolar

```
1
2  //SUCCESS_TEST_BIPOLAR
3  //Modified version of success_test for bipolar
4  //electrode configurations. See success_test
5  //for original test file and how it works.
6  //This file is modified to ignore the
7  //expected delay test as it caused problems
8  //with bipolar blocking electrode configurations
9  //outputting many false negatives since the
10 //larger blocked area slowed down incoming
11 //action potentials significantly, which
12 //confused the timing test.
13 //The numbers of passing action potentials
14 //are still counted however and serve to
15 //determine if selective stimulation occurred
16
17 //Arguments
18 //$o1 = list of apcounter time vectors for small axon
19 //$o2 = list of apcounter time vectors for large axon
20 //$o3 = matrix of results
21 //$o4 = vector for variable 1 values (x values)
22 //$o5 = vector for variable 2 values (y values)
23 //$6 = value of variable 1 at which the test is made (index of current x in x value vector)
24 //$7 = value of variable 2 at which the test is made (index of current y in y value vector)
25 //$8 = onset response delay in [ms]
26
27 proc success_test_bipolar() {local small_axon_APpropagation_speed, small_APCounter_distance,
       expected_delay, i localobj apcount_timevec_proximal_small, apcount_timevec_distal_small,
        apcount_timevec_proximal_large, apcount_timevec_distal_large
28
29   //Remove any ignored values within the vectors recording detected APs
30   apcount_timevec_proximal_small = new Vector()
31   apcount_timevec_distal_small = new Vector()
32   apcount_timevec_proximal_large = new Vector()
33   apcount_timevec_distal_large = new Vector()
34
35   if ($o1.o(0).size()>=1 && $o1.o(1).size()>=1) {
36     for i=0,$o1.o(0).size()-1 {
37       if ($o1.o(0).x[i] > $8) {
38         apcount_timevec_proximal_small.append($o1.o(0).x[i])
39       }
40
41     }
42     for i=0,$o1.o(1).size()-1 {
43       if ($o1.o(1).x[i] > $8) {
44         apcount_timevec_distal_small.append($o1.o(1).x[i])
45       }
46     }
47     for i=0,$o2.o(0).size()-1 {
48       if ($o2.o(0).x[i] > $8) {
49         apcount_timevec_proximal_large.append($o2.o(0).x[i])
50       }
51
52     }
53     for i=0,$o2.o(1).size()-1 {
54       if ($o2.o(1).x[i] > $8) {
55         apcount_timevec_distal_large.append($o2.o(1).x[i])
56       }
57     }
58
59
60     if (apcount_timevec_proximal_small.size()==1 && apcount_timevec_distal_small.size()==1 &&
        apcount_timevec_proximal_large.size()==1 && apcount_timevec_distal_large.size()==0) {
61
62         $o3.x[$6][$7]=1
63         print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed
        selective stim was successful."
64
65
66     } else {
67       $o3.x[$6][$7]=0
68       print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed selective
       stim failed.\nReason: AP count mismatch.\nCode: ",apcount_timevec_proximal_small.size(),
       apcount_timevec_distal_small.size(),apcount_timevec_proximal_large.size(),
       apcount_timevec_distal_large.size()
69     }
70   } else {
71     $o3.x[$6][$7]=0
72     print "at x value ", $o4.x[$6], "and y value ", $o5.x[$7], "\nThe test showed selective
       stim failed.\nReason: APs annihilated/missing APs"
73   }
74 }
```

### block_test_singleaxon

```

//BLOCK_TEST
//Function returns boolean using a test to determine
//if nerve block really occurred during the
//simulation. It aims at filtering out cases where:
//1. Block was partial/inexistent, leading to generation
//of additional APs at block location
//2. Stimulated AP made it through the blocked region

//If none of the above are true then the test returns 1 (true)
//else it returns false.

//APCounter time vectors contain the times in the simulation
//at which the counters were triggered. By using logic it
//is possible to test for cases where nerve block was
//achieved, categorizing any other case as a failure.
//Taking into account the onset response time parameter,
//all recorded APs before a certain time are ignored (the
//values are removed from the vectors used for the test
//logic). This time is calculated by adding the onset
//response time to the time necessary for an action potential
//to propagate from one end of the axon to the other; in
//this way it is ensured that all APs that could have been
//generated during the expected onset response are ignored.
//For the rest of the detected action potentials, there has
//to be num_pulse action potentials detected proximally,
//and 0 action potentials detected distally.


//Arguments
//$o1 = list of apcounter time vectors for the axon
//$o2 = matrix of results
//$o3 = vector for variable 1 values (x values)
//$o4 = vector for variable 2 values (y values)
//$5 = value of variable 1 at which the test is made (index of current x in x value vector)
//$6 = value of variable 2 at which the test is made (index of current y in y value vector)
//$7 = number of action potentials sent by the stimulatory electrode (num_pulse)
//$8 = onset response delay in [ms]


proc block_test() { local i localobj apcount_timevec_proximal, apcount_timevec_distal

  //Remove any ignored values within the vectors recording detected APs


  apcount_timevec_proximal = new Vector()
  apcount_timevec_distal = new Vector()

  if ($o1.o(0).size()>=2 && $o1.o(1).size()>=1) {
    for i=0,$o1.o(0).size()-1 {
      if ($o1.o(0).x[i] > $8) {
        apcount_timevec_proximal.append($o1.o(0).x[i])
      }

    }
    for i=0,$o1.o(1).size()-1 {
      if ($o1.o(1).x[i] > $8) {
        apcount_timevec_distal.append($o1.o(1).x[i])
      }
    }
    apcount_timevec_proximal.printf()
    apcount_timevec_distal.printf()
    if (apcount_timevec_proximal.size()==$7 && apcount_timevec_distal.size()==0) {

        $o2.x[$5][$6]=1
        print "at x value ", $o3.x[$5], "and y value ", $o4.x[$6], "\nThe test showed that the
       nerve block was successful."

    } else {
      $o2.x[$5][$6]=0
      print "at x value ", $o3.x[$5], "and y value ", $o4.x[$6], "\nThe test showed that the
      nerve block failed.\nReason: AP count mismatch.\nCode: ",apcount_timevec_proximal.size()
      ,apcount_timevec_distal.size()
    }
  } else {
    $o2.x[$5][$6]=0
    print "at x value ", $o3.x[$5], "and y value ", $o4.x[$6], "\nThe test showed that the
     nerve block failed.\nReason: APs annihilated/missing APs"
  }
}
```

### ap_speed_measure

```

```

```
2  //AP_SPEED_MEASURE
3  //Function for the measurement of action potential
4  //speeds based on AP counter data. The test supposes only
5  //one AP was generated and two APcounters on each axon.
6  //Distance between the APcounters is calculated from node
7  //indexes and known section lengths (node and internode).
8  //This procedure can be used to calculate speeds using any
9  //combination of 2 separate APCounters on a single axon.
10 //Care must be taken not to use APCounters that could be
11 //perturbed or confused by any phenomena happening at their
12 //site i.e. blocking signals. Do not use APCounters at
13 //blocking sites if blocking signals are used in tests as
14 //the blocking waveform will repeatedly trigger the
15 //APCounter and lead to erroneous measurements.
16 //The function can return negative speeds depending on the
17 //direction of propagation of the action potential, and the
18 //order in which the APCounters are specified.
19
20 //Arguments:
21 //$o1 = list of apcounter time vectors
22 //$2 = index of 1st APCounter's time vector in list
23 //$3 = index of 2nd APCounter's time vector in list
24 //$4 = index of node containing 1st APCounter
25 //$5 = index of node containing 2nd APCounter
26 //$6 = total internode length (node length + internode lengths)
27
28
29 func ap_speed_measure() {local time_start, time_end, dist
30   time_start = $o1.o($2).x[0]/1000  //Conversion from ms to s
31   time_end = $o1.o($3).x[0]/1000    //Conversion from ms to s
32   dist = ($5-$4)*$6/1000000     //Conversion from microns to meters
33   return (dist/(time_end-time_start)) // [m.s^-1]
34 }
```

All functions have an integrated manual explaining their use.