BACHELOR' S THESIS / BIOMEDICAL ENGINEERING 2022

Development of Skeletal Muscle Contraction Computer Physiology Practices for Human Biology, Medicine and Biomedical Engineering

Karina Karenik

upf. Universitat Pompeu Fabra Barcelona

Development of Skeletal Muscle Contraction Computer Physiology Practices for Human Biology, Medicine and Biomedical Engineering

Karina Karenik

Bachelor's thesis UPF 2021/2022

Thesis supervisor(s):

Dr. José M. Fernández Fernández

Department of Medicine and Life Sciences

Patricia Carbajo Flores

Universitat Pompeu Fabra Barcelona

Acknowledgments

Firstly, I would like to thank my supervisors, Dr. José M. Fernández Fernández and all of the colleagues in the department as well as Patricia Carbajo Flores, for giving me the opportunity to perform this Bachelor Thesis and for the constant help and guidance throughout the process.

To my family, for encouraging me in taking Biomedical Engineering degree in a foreign country, without knowing the local languages and making me jump completely outside my comfort zone. To all the people around me who supported and believed in me or helped me in so many different ways until the very end for the past years.

Abstract

Inside Pompeu Fabra University students of Biomedical Engineering, Medicine and Human Biology bachelor degrees must perform several physiology practices by using a software. In more detail, the software in question is related to the skeletal muscle contraction in response to electrical stimulations of the motor nerve.

During the last years a clear need for better performing software has been identified due the incompatibility with current operating systems and visual aspects. Therefore, this project proposes and designs a new version of an open-source software that covers the need of physiology practices for the bachelor students. The project was carried out covering the three main subjects related to action potential of skeletal muscle, summation of motor units and wave summation that leads to tetanization. It was possible to carry out the design of the program thanks to Python programming language and the use of PyQt package along with its' functions.

Keywords

Skeletal muscle contraction, computer physiology practices, action potential, summation of motor units, wave summation, tetanization, python, PyQt

Index

1. Introduction

A muscle is a group of human body muscle tissues which contract together and allow the movement of our body thanks to the contractions and relaxations of their substructures. Muscles can be classified having into account their cellular composition and function. These can be of three types: cardiac, smooth, or skeletal[1]. Cardiac muscle is found in the heart and is responsible for the blood pumping though it by contraction. Smooth muscle can be found in the walls of hollow internal structures such as blood vessels, uterus or gallbladder. Skeletal muscle is responsible for the axial and appendicular movements of the body providing it stability while performing daily life actions.

The skeletal muscle system is distributed throughout the body and is anchored to the bones by tendons. The voluntary nervous system, or somatic nervous system, is part of the peripheral nervous system which controls the voluntary control of body movements by using skeletal muscles. The central nervous system (CNS) is connected to the voluntary nervous system through sensory nerves carrying afferent nerve fibres[2]. Additionally, the somatic nervous system also consists of motor nerves that carry efferent nerve fibres. Motor nerves are responsible for conducting the neuronal signals that eventually reach the muscle in order to produce its contraction. In the Fig.1, we can observe general structure of the skeletal muscle.

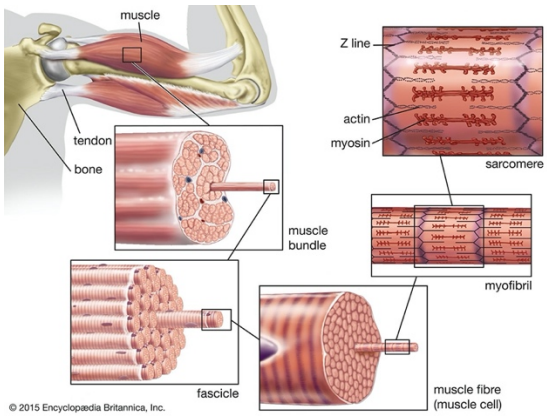

Fig.1: Structure of skeletal muscle

Skeletal muscle is formed of numerous fibres of about 10 to 80 μm in diameter, which are composed of myofibrils. These myofibrils are made of thin actin and thick myosin filaments and are surrounded by a plasma membrane called sarcolemma. Sarcomere is the contraction unit of the muscle fibres and is defined as the segment between two neighbouring Z-lines. If we take a closer look at the electron micrograph of a skeletal muscle sarcomere (Fig.2), we are able to see the thick and thin filaments and the banding pattern (bands H, A and I) of the muscle fibre[3].

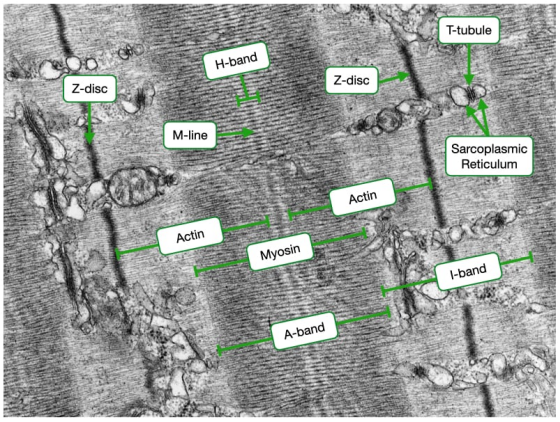

Fig.2: Electron micrograph of a skeletal muscle sarcomere

The motor unit consists of the somatic motor neuron together with the sarcolemma and its myofibrils. Once the electrical impulse reaches the neuronal axon, the voltage-gated calcium channels are activated allowing calcium ions enter inside the neuron. These calcium ions bind to their receptors on the synaptic vesicles, triggering the fusion of the vesicles with the plasma membrane of the axon and causing the acetylcholine to be released into the synaptic cleft. After binding to the cholinergic receptors on the sarcolemma the myofibrils are contracted.

1.1. Skeletal Muscle Action Potential

The resting potential of a skeletal muscle cell is of -90 mV. On average the action potential has a duration between 1 and 5 msec. and has a propagation velocity of around 3 - 5 m/s. The specialized structure that is able to transmit the action potential throughout the fibre is called Transverse tubules (T-tubules). These are extensions of the cell membrane that penetrate into the centre of skeletal muscle cells. They are located very close to the membrane of the sarcoplasmic reticulum which releases calcium (Ca^{2+}) as a response to the action potential as it activates the voltage-gated calcium channels located at T-tubule membrane. Due high availability of ATP the displacement of myosin filaments over actin filaments occurs. The calcium attaches to the troponin and the

contraction of the sarcomere begins. The movements of myosin can be described as a kind of molecular choreographed dance. The myosin reaches forward, binds to actin, contracts, releases actin, and then reaches forward again to bind actin once again in a new cycle. This process is called myosin-actin cycling. As the myosin S1 segment binds and releases actin, it forms cross bridges, which extend from the thick myosin filaments to the thin actin filaments. The contraction of myosin S1 region is called the power stroke (Fig. 3). The power stroke requires the hydrolysis of ATP, which breaks a high-energy phosphate bond to release energy[4].

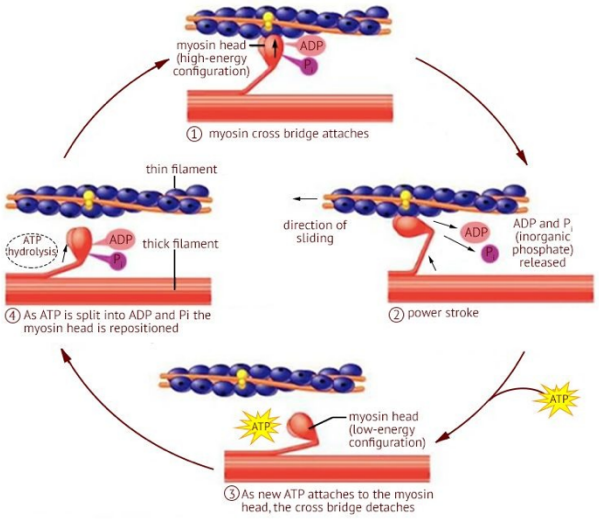

Fig. 3: Myosin-actin cycle

As the thin filaments are anchored to the Z line, the sliding of the filaments causes each sarcomere, and therefore the muscle fibres, to shorten.

Once calcium ions have been released, they must be removed for the contraction process to cease. For that process in the membrane of the reticulum there is a calcium ATPase that introduces the Ca^{2+} back into the reticulum.

1.2. Summation of motor units

The contraction of each muscle fibre will produce the contraction of the muscle by a summation process. When a small contraction is desired, only a few cells contract at a time. However when a stronger contraction is needed, a large number of fibres contract at the same time. In general, the different degrees of contraction of a muscle are achieved by two types of summation: summation of multiple motor units and summation of

waves[6]. The force of contraction of a muscle increases progressively when the number of motor units recruited increases. In each muscle, the number of muscle fibres and their size vary from one motor unit to another, so that one motor unit can be much stronger than others. Small motor units are more easily excitable as they are innervated by small nerves which are easily excitable from the spinal cord. Therefore, the contraction of these motor units is very fast. On the other hand, if motor units with large amount of fibres are involved, the contraction is slower.

1.3. Wave summation and tetanization

When a muscle is stimulated with a progressively increasing frequency, there comes a time when the successive contractions merge into one another and can no longer be distinguished individually. This state is known as tetanization, tetanus or tetanized state. The threshold frequency at which tetanus occurs is called the critical frequency. Once this critical frequency is reached, an increase in the stimuli induces a very weak increase in the force of contraction and therefore the system loses the linearity in its response [5] (Fig.4).

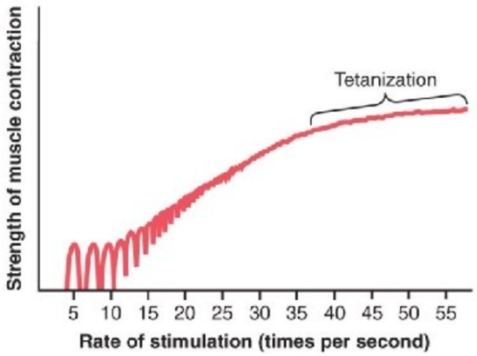

Fig. 4: Wave summation and eventual tetanization

2. State of the art

In an academic environment the study and understanding of human physiological systems in general currently include theoretical contents and lectures as well as some seminars. Practical and laboratory sessions have an important role as well in courses like Physiology in Human Biology, Medicine and Biomedical Engineering degrees. In order to be able to practice and learn different concepts in a more interactive way, arises the need of a mathematical computerized model that allows us to investigate different aspects of muscle contraction.

One of the software that is being currently used at Universidad Pompeu Fabra in Physiology course related to muscle contraction and the study of the concepts described earlier, is "*The Frog Muscle Lab*" designed by Philip Blake and Dr. Timothy Simons from Kings's College of London. This software was designed to replace laboratory sessions, which involved the use of animals, more specifically frogs, as the experiments are related to the contraction of the sciatic nerve from *ex vivo* dissected frog.

Students' continuing pedagogic needs are the centre of the design of this software and at the same time they tried to retain the inherently variable nature of laboratory work. The data and algorithms used by the software were generated from actual experiments, and are subject to realistic randomisation to ensure that students' laboratory results and experiences are different. Nevertheless, the software was designed more than two decades ago which leaves a lot of room for improvement regarding the Graphic User Interface and compatibility with current systems.

3. Objectives

The aim of this project is to design a new version of an open-source software that allows to simulate the skeletal muscle contraction in response to electrical stimulation of the motor nerve. The software's main users are focused to be Pompeu Fabra University students of Biomedical Engineering, Medicine and Human Biology degrees during the physiology practices.

Having in mind that the previous version of the software used up to date only operates in Windows 98, the software developed during this project should work in most current operating systems.

The purpose of the proposed simulations is to be able to study in more detail and experimentally explain several concepts related to skeletal muscle contraction such as:

- Recruitment or summation of motor units in response to a single electrical stimulus of increasing amplitude;

- Temporal and sequential representation of electrical stimulation of the motor nerve, action potential in the muscle and muscle contraction;

- Summation of contraction waves and tetanisation as a consequence of the application of high amplitude stimuli at increasing frequencies;

- Relationship between the length of a skeletal muscle and its contractile capacity after tetanus stimulation.

4. Materials & Methods

4.1. Technical Description

The current version of the software of Physiology Practices was programmed from scratch in Python programming language (version 3.9.5) [7] making use of PyQt package for the Graphic User Interface (GUI). Coding files (.py format) were generated, coded and modified with the software Spyder (version 4.1) [8]. Spyder is a free and open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language.

On the other hand, PyQt package [9] gives the freedom to create GUIs of my choice while also providing a lot of good pre-built designs and widgets. By using Qt Designer software (version 5.10.0), each of the windows, buttons, tables and all type of widgets of the software were manually designed, defining the sizes, colours, many visual aspects and the distribution of the contents in each of them. This tool provides us with a what-you-see-is-what-you-get (WYSIWYG) user interfaces. Later on, each of these windows that were saved in .ui extension, were linked to the corresponding .py file containing the technical function of those program windows, where the performance of each button was programmed. Each of the files (.ui files and .py files) of each window had to be indexed and connected together in the main menu.

All of the files, necessary data and code folders were uploaded to GitHub, which is a code hosting platform for version control and collaboration [10]. The information needed related to the code can be found here: https://github.com/KarinaKarenik/Skeletal-Muscle-Contraction-Lab.git .

Furthermore, to be able to change visual aspects of the program all at once, a .qss file [11] was generated which can be edited in any text processing program. In other words, these visual characteristics are called Styles and are very commonly used in Qt. QSS file was defined in order to reduce coupling and also separate it from the logical code. Within it, the styles of various components (such as QLable, QLineEdit, QPushButton, etc.) were written and then it was finally used QApplication to load the styles, so that the whole application shares the same style within it.

All the necessary files to run the program where organized into folders as follows:

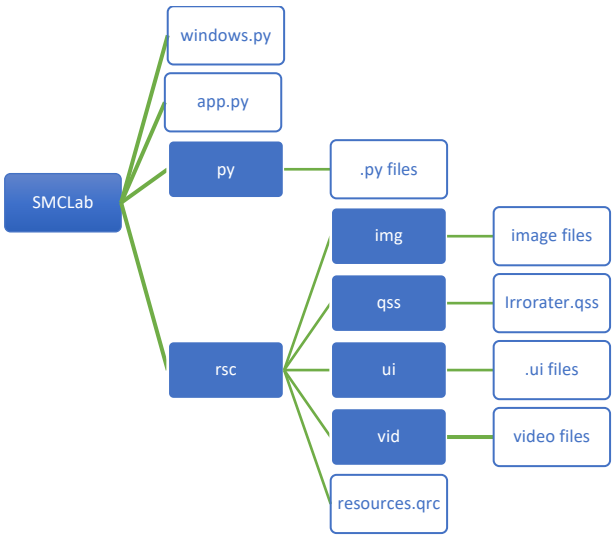

Fig. 5: Organization of coding files

Throughout all of the .py files generated several packages and functions such as QtWidgets, random, loadUi, QPixMap, pyqtgraph had to be imported (Fig.6).

```
from PyQt5 import QtWidgets
from PyQt5.uic import loadUi
from PyQt5.QtGui import QPixmap
import pyqtgraph as pg
import random, windows
```

Fig. 6: Imported packages and functions

All of the windows of the program are defined inside a class and indexed so they follow a chronological and logical order when a button is pressed (Fig.7). The software has a total of 28 windows that are distributed between different sections explained above.

```
class DataWindows:
    windows = [py.Menu.Menu(), #0
    py.Theory1.Theory1(),      #1
    py.Theory2.Theory2(),      #2
    py.Theory3.Theory3(),      #3
    py.Theory4.Theory4(),      #4

    py.Experiment1_one.Experiment1_one(),  #5
    py.Experiment1_two.Experiment1_two(),  #6
    py.Experiment1_three.Experiment1_three(),  #7
    py.Experiment1_four.Experiment1_four(), #8
    py.Experiment1_five.Experiment1_five(), #9

    py.Experiment2_one.Experiment2_one(),   #10
    py.Experiment2_two.Experiment2_two(),   #11
    py.Experiment2_three.Experiment2_three(),  #12
    py.Experiment2_four.Experiment2_four(),    #13
    py.Experiment2_five.Experiment2_five(),    #14

    py.Experiment3_one.Experiment3_one(), #15
    py.Experiment3_two.Experiment3_two(), #16
    py.Experiment3_three.Experiment3_three(), #17
    py.Experiment3_four.Experiment3_four(), #18
    py.Experiment3_five.Experiment3_five(), #19


    py.Experiment4_1_one.Experiment4_1_one(), #20
    py.Experiment4_1_two.Experiment4_1_two(), #21
    py.Experiment4_1_three.Experiment4_1_three(), #22
    py.Experiment4_2_one.Experiment4_2_one(), #23
    py.Experiment4_2_two.Experiment4_2_two(), #24
    py.Experiment4_2_three.Experiment4_2_three(), #25
    py.Experiment4_conclussions.Experiment4_conclussions(), #26

    py.Conclussions.Conclussions() #27
    ]

    windows[6].loadChart(windows[7])
    windows[23].loadChart(windows[24])


    widgets = QtWidgets.QStackedWidget()
    widgets.setFixedWidth(1000)
    widgets.setFixedHeight(750)

    for win in windows:
        widgets.addWidget(win)
```

Fig. 7: Windows .py file with all the windows indexed

General buttons such as “Next”, “Back” and “Menu” are programmed in the same way in all of the windows by the use of functions defined inside the corresponding .py files (Fig.8).

```
def goBack(self):
    windows.DataWindows.widgets.setCurrentIndex(windows.DataWindows.widgets.currentIndex()-1)

def goNext(self):
    windows.DataWindows.widgets.setCurrentIndex(windows.DataWindows.widgets.currentIndex()+1)

def goToMenu(self):
    windows.DataWindows.widgets.setCurrentIndex(0)
```

Fig. 8: “Next”, “Back” and “Menu” buttons coding

4.2. Visual and colour set-up

The visual representation of the software was thought to be as simple and as user-friendly as possible, extracting all the unnecessary buttons or information. The idea was to keep it as simple as possible.

On the other hand, when designing the visual impact and colour choice, I had to take into account that the colours used should be colour-blind friendly. That is, avoid the use of red and green together. Let’s note that the colour that looks the most the same for

people with normal vision and readers with red-/green-blindness (the most common types of colour-blindness) is blue. Said this, the colours of choice for the software were mainly shadows of blue, with some details, such as scroll bar or when selecting next in green and some of the next in orange, to create contrast. Very often, we can observe that blue/orange is a common colourblind-friendly palette [12].

4.3. Data and Materials

In the best case scenario, the optimal situation would have been gathering the access to the original coding files and data from the software "*The Frog Muscle Lab"* from King's College London. Nevertheless, due the time that have passed since (more than 20 years) and the format used, direct access to the data was impossible to accomplish. Therefore, a different approach had to be taken in order to be able to replicate the results.

The experimental data and algorithms for different experiments had to be extracted manually from the original software of "*The Frog Muscle Lab",* with the consent from King's College and original authors. Said this, the new version of the software developed in this project also refers to the contraction of the *ex vivo* frog muscle using the sciatic nerve.

In more detail, for each section of the software, different type of data had to be introduced and explained:

Theory section: General and brief information as well as visual representations such as images and videos related to the dissection of the frog and the sciatic nerve as well as the experimental hardware set-up.

Experiment 1: Voltage (V) (from 0 to 4 Volts) and Peak Force (g) (from 0 to 142 g) data to be plotted that is obtained when stimulating the sciatic nerve with single shocks of different voltage of fixed duration (0.5 msec). The nerve was stimulated while the muscle was held under isometric conditions that is, at constant length.

Experiment 2: Time (ms) (from 0 to 150 ms) and Tension (g weight) (from 0 to 45 g weight) of stimulus, action potential and contraction data when stimulating the muscle with a single supramaximal shock. The stimulus, muscle action potential and muscle contraction have been recorded simultaneously using a 3-channel recorder. The muscle contraction is calibrated in g weight, but the action potential was recorded using arbitrary scale.

Experiment 3: Time (ms) and Force (g) data of muscle contractions of repetitive stimulations for 2 seconds with a frequency set to any value between 1 and 50 stimuli per second. Each stimulus was supramaximal, of 4 Volts, and 0.5 msec duration.

Experiment 4: Tension (g weight) (from 0 to 400 g weight) and Time (sec) (from 0 to 5 sec) data while investigating the effect of varying the length of the muscle (mm) (from 25,5 to 35 mm) adjusting it by means of a screw thread in the stand. In this experiment, the muscle is always stimulated tetanically, using supra-maximal stimuli, so it develops the maximum possible force. Also, data obtained of the resting tension (g weight) (from 0 to 330 g weight) and the tetanic tension (g weight) (from 36,5 to 375,5 g weight) when changing the length of the muscle.

4.4. Methods

The software was thought to have a main menu from which we can select several options. The main parts of the software are the theory section along with the four different experiments that explain different physiological situations (Fig.9).

Fig. 9: Main menu options

Each button of the main menu was coded separately in a function, in order to separate the logical performance (Fig.10).

```
class Menu(QtWidgets.QMainWindow):
    def __init__(self):
        super(Menu, self).__init__()
        loadUi("./rsc/ui/menu.ui",self)
        self.setWindowTitle('Menu')
        self.pushButtonTheoryMenu.clicked.connect(self.goToTheory)
        self.pushButtonExperiment1Menu.clicked.connect(self.goToExperiment1)
        self.pushButtonExperiment2Menu.clicked.connect(self.goToExperiment2)
        self.pushButtonExperiment3Menu.clicked.connect(self.goToExperiment3)
        self.pushButtonExperiment4Menu.clicked.connect(self.goToExperiment4)
        self.pushButtonConclusion.clicked.connect(self.goToConclusion)
        self.pushButtonAboutMenu.clicked.connect(self.showAbout)
        self.about = About(self)
```

Fig. 10: Coding of each button in main menu

The user must complete each experiment/exercise and reply to several mandatory questions correctly in order to be able to jump to the next experiment. The temporal length of the software should take no longer than one hour and a half, which is the time slot that is usually assigned to students for this type of practices.

Each of the main sections of the new software is described in more detail below.

4.4.1 Theory section

The frog used in this Lab is a North American leopard frog, *Rana pipiens*, which is slightly larger than the common frog, *Rana temporaria*.[13] In order to proceed with the experimentations, the animals must be anesthetized or killed before we can perform experiments on them. In our case, as per usual procedure, the frog is killed by a process called pithing. The frog is stunned by a blow to the head, then a metal probe is inserted at the back of the neck to destroy the brain and spinal cord. The central nervous system has now been destroyed, so the frog can feel no pain. The dissection is performed with care in order to avoid cutting large blood vessels. The most important similarity between frog and mammalian physiology is that the nerve and muscle systems work in the same way.

Frog tissues survive best if they are bathed in a solution (Ringer Frog solution) [14] with approximately the same salt composition as frog blood plasma. Experiments with frog tissue are carried out at room temperature as this type of frog lives at environmental temperatures of 5º - 25ºC.

The preparation consists of the gastrocnemius muscle [15], which is the large muscle at the back of the lower leg, plus the sciatic nerve which controls it. The muscle has to be dissected with the nerve attached and care is needed to preserve the sciatic nerve roots, which are three on each side of the spinal cord. The sciatic nerve is dissected from its roots in the pelvis, down to the knee. Finally the upper leg is cut above the knee joint, leaving the isolated sciatic nerve-gastrocnemius muscle preparation.

The set-up of the experiment follows several steps as described:

1- The frog sciatic nerve-gastrocnemius muscle preparation is mounted in the muscle bath;

2- The nerve is stimulated with electric impulses from the stimulator;

3- When the muscle contracts this is detected by the strain gauge;

4- The strain gauge sends an electric current to the interface, which converts it to a digital signal, which appears on the computer screen;
5- The computer records these signals for analysis later on.

The stimulator produces a series of pulses, which can be varied in voltage from 0.1 volts to 100 volts. The duration of each pulse can vary from 0.02 msec to 200 msec. One can give either a single pulse or repeat pulses, at any frequency up to 200 Hz. Each pulse consists of a negative step impulse of direct current.

The knee joint is placed in the nerve compartment, so that it fixes the muscle at one end. The other end will be fixed by the thread attached to the Achilles tendon. The preparation is attached to a stand. The thread around Achilles tendon is attached to the strain gauge[16], which will measure the tension generated by the muscle. The muscle bath is rotated so it is vertical, and placed in a beaker of Frog Ringer.

The strain gauge contains a piezo-electric crystal which generates a voltage when a force is applied to it. This happens when the muscle pulls on the hook. The interface converts the signal to a digital form and plots it as a graph of voltage (y-axis) against time (x-axis). Red channel shows muscle tension (tension units of gram weight) and the green channel shows electrical activity in the muscle (in mV). Furthermore, it is important to calibrate the apparatus, in order to relate the signals on the computer to the force generated by the muscle.

4.4.1.1 Coding files

In order to code the corresponding section of the software a total of four .py files were used, one for each window. All of them contain similar structure and each file is connected to the corresponding .ui file which contain all the widgets (Fig.11).

```
class Theory2(QtWidgets.QMainWindow):

    def __init__(self):
        super(Theory2, self).__init__()
        ui_path = "rsc/ui/theory2_preparation.ui"
```

Fig. 11: Code for connecting respective .ui file

Text information from these windows were manually introduced in .ui files at the corresponding textboxes. However, in order to be able to reproduce the video that shows the dissection of the frog on the second window (Theory2.py file), a special code had to be introduced (Fig.12) and the video, which originally was in .mp4 format had to be converted into .gif to avoid problems in reproducing it.

```
filepath = "rsc/vid/video1.gif"
if getattr(sys, 'frozen', False):
    ui_path = os.path.join(sys._MEIPASS, ui_path)
    filepath = os.path.join(sys._MEIPASS, filepath)
loadUi(ui_path,self)
#self.setup()
self.movie = QtGui.QMovie(filepath)
self.label_2.setMovie(self.movie)
self.movie.start()
```

Fig. 12: Incorporation of video file

4.4.2 Experiment 1

In this experiment we stimulate the sciatic nerve with single shocks of fixed duration (0.5 msec) and increasing voltage. The shock is applied as a negative square wave pulse to the electrode that is closer to the muscle (Fig.13)

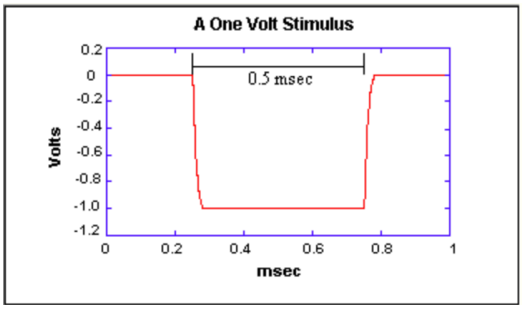

Fig. 13: Single shock stimulation

The experiment is performed following the instructions and the user is asked the following questions:

- Start at a low stimulation voltage, and steadily increase the voltage.
- What is the minimum stimulus needed to produce a response?
- As you increase the stimulus strength, does the response reach a maximum?

Finally, the results are plotted we can see how peak force varies with voltage.

At the end of the experiment, the user is asked five questions that must be replied:

1- From your results, what is the minimum voltage required to elicit a response?

2- The sciatic nerve of the frog contains a number of sensory and motor nerve fibres. Complete the following statement about threshold voltage from the word choices under each text box.

3- This is the minimum voltage required to stimulate one/all of the sensory/motor fibres.

4- If you stimulate twice with the same voltage, do you get the same response?

5- Stimulation of the sciatic nerve with larger and larger voltages above threshold results in larger and larger contractions. This process is called recruitment. Each individual motor nerve fibre has a different threshold, because some are further away from the stimulating

electrode than others. As the voltage is increased, more motor nerve fibres are stimulated, until eventually they are all stimulated. When this occurs, all the muscle fibres contract simultaneously, resulting in maximal response. From your results: What is the approximate average value of the maximum response?

6- Is recruitment compatible with all-or-nothing law?

Finally, as a conclusion, we can say that in this experiment we have seen that there is a threshold, or minimum voltage, required to stimulate each nerve fibre. The sciatic nerve consists of a bundle of fibres, with a range of thresholds. A large stimulus that activates all the fibres is called a supramaximal stimulus. When a nerve fibre is stimulated, an action potential passes down the nerve, causes release of the transmitter, and activates the muscle. The activation of the muscle results first in an action potential in the muscle cell membrane, and secondly in a contraction, or twitch. In this experiment we have observed the twitch.

4.4.2.1 Coding files

In this first experiment, just as in theory section, all of the windows, which are five in total, share similar code but also contain specific code.

In order to be able to save the values of voltage applied (between 1 and 4 Volts), chosen by the user, and force obtained within a table the buttons that generate and apply the voltage had to be programmed according to the values of force expected within the biological sense (Fig.14):

```
def valueChanged(self):
    value = self.dial.value()
    value = value / 1000
    self.lcdNumber.display(str(value))

def applyVoltage(self):
    force = 0
    value = self.dial.value()
    value = value / 1000
    if (value >= 2.4):
        force = random.randint(117,142)
    elif (value >= 2.3):
        force = random.randint(117,140)
    elif (value >= 2.0):
        force = random.randint(114,135)
    elif (value >= 1.9):
        force = random.randint(107,129)
    elif (value >= 1.8):
        force = random.randint(111,123)
    elif (value >= 1.7):
        force = random.randint(101,116)
    elif (value >= 1.6):
        force = random.randint(85,99)
    elif (value >= 1.5):
        force = random.randint(67,81)
    elif (value >= 1.4):
        force = random.randint(49,56)
    elif (value >= 1.3):
        force = random.randint(26,29)
    elif (value >= 1.2):
        force = random.randint(10,12)
    elif (value >= 1.1):
        force = 2
    else:
        force = 0
    rowPosition = self.tableWidget.rowCount()
    self.tableWidget.insertRow(rowPosition)
    self.tableWidget.setItem(rowPosition,0,QtWidgets.QTableWidgetItem(""+str(value)))
    self.tableWidget.setItem(rowPosition,1,QtWidgets.QTableWidgetItem(""+str(force)))
```

Fig. 14: Values of force to be obtain according to voltage applied

Next, the values that were generated by the user had to be reproduced within a plot in scattered matter for better visualization. For that, a UiComponents class had to be created, and the plot introduced within a gridLayout (Fig.15).

```
def clearLayout(self):
    for i in reversed(range(self.gridLayout.count())):
        widgetToRemove = self.gridLayout.itemAt(i).widget()
        # remove it from the layout list
        self.gridLayout.removeWidget(widgetToRemove)
        # remove it from the gui
        widgetToRemove.setParent(None)

def UiComponents(self):
    self.clearLayout()
    self.gridLayout
    plot = pg.plot()
    # creating a scatter plot item
    # of size = 10
    # using brush to enlarge the of green color
    scatter = pg.ScatterPlotItem(size=10, brush=pg.mkBrush(30, 255, 35, 255))
    # adding spots to the scatter plot
    scatter.addPoints(self.x_data, self.y_data)
    plot.addItem(scatter)
    # set properties of the label for y axis
    plot.setLabel('left', 'Peak force', units ='g')
    # set properties of the label for x axis
    plot.setLabel('bottom', 'Voltage', units ='V')
    self.gridLayout.addWidget(plot)
```

Fig. 15: Plot and gridLayout definition

In the last part of this exercise, when the user is asked several questions, these had to be coded having in mind that the correct answers should be contained within a value range (Fig.16).

```
def answerQuestion1(self):
    x = float(self.lineEditQuestion1.text())
    if (x > 1.4):
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! The voltage is TOO BIG")
    elif (x > 0.5):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! That is the threshold voltage.")
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! At that voltage you have NO response.")

def answerQuestion2(self):
    if (self.comboBox.currentText() == "one" and self.comboBox_2.currentText() == "motor"):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! A voltage just above the threshold will stimulate one mo
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect! A voltage just above the threshold will stimulate on

def question3Yes(self):
    QtWidgets.QMessageBox.about(self, "Incorrect!", "No, you are Incorrect!")

def question3No(self):
    QtWidgets.QMessageBox.about(self, "Correct!", "Correct! The frog nerve-muscle preparation shows biological variabili

def answerQuestion4(self):
    x = float(self.lineEditQuestion4.text())
    if (x > 136):
         QtWidgets.QMessageBox.about(self, "Incorrect!", "Your answer is too big. Go back, make more observations, and u
    elif (x > 119):
         QtWidgets.QMessageBox.about(self, "Correct!", "Well done.")
    else:
         QtWidgets.QMessageBox.about(self, "Incorrect!", "Your answer is too small. Go back, make more observations, and

def question5Yes(self):
    QtWidgets.QMessageBox.about(self, "Correct!", "Correct! This law applies to each individual motor nerve fiber in the

def question5No(self):
    QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect")
```

Fig. 16: Questions and answer conditions

4.4.3 Experiment 2

In this experiment we stimulate the muscle with a single supramaximal shock, and observe both the contraction of the muscle, and the action potential in the muscle fibres.

The action potential is recorded with an extracellular electrode just touching the muscle, as shown in the diagram (Fig.17).

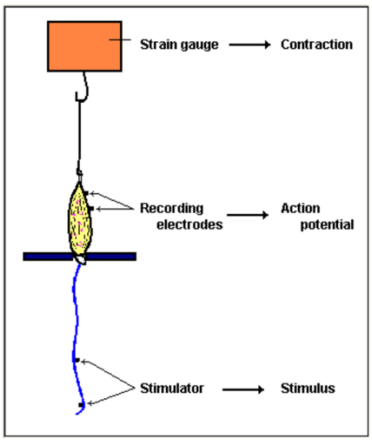

Fig. 17: Diagram of electrode touching the muscle

When pressing the button of stimulation, we obtain a graph where the stimulus, muscle action potential and muscle contraction have been recorded simultaneously using a 3-channel recorder.

After observing the graph, the user is asked the following question, before proceeding to the next part: How long does the contraction last (in msec)?

The next page of the software shows a portion of the experiment on the previous page, with the time scale expanded. It is important to note that the action potential in the muscle comes before the contraction. In this section of the lab the user is also asked some questions:

1- What is the delay between the stimulus and the start of the muscle action potential?

2- What is the delay between the beginning of the muscle action potential and the start of the muscle contraction?

At the end, we can say that in this experiment we have stimulated the nerve with a single supramaximal shock, and observed the generation of an action potential, and a single contraction in the muscle. We are able to see that the contraction started about 6 msec after the stimulus, and lasted for approximately 120 msec. The action potential in the muscle precedes the contraction, and occurs about 1 msec after the stimulus to the nerve. The interpretation of the experiment is affected by two factors:

- The time resolution of the apparatus was 1 msec, so this is the smallest time difference that could be measured.

- A bundle of nerves was stimulated. Each axon in the bundle supplies a motor unit, a group of muscle fibres. The conduction velocity of each axon is slightly different, so the different motor units in the muscle are all stimulated at slightly different times.

The start of the muscle action potential shows the time of the action potential in the first motor unit to be stimulated, and the start of the contraction shows the time at which the contraction starts in the first motor unit. The total time taken for the action potential or the contraction includes the spread of time between the activation of different motor units.

4.4.3.1 Coding files

During the second exercise, on the second window, the user is asked to indicate in msec the value that corresponds to the contraction that be extracted from to the graph presented. In order to do so, the user is given two vertical bars which can be moved horizontally and therefore obtain the value that corresponds to the difference between the final and initial bar positions (Fig.18).

```
def answerQuestionOne(self):
    x = float(self.lineEditQuestion1.text())
    if (x >= 114 and x <= 116):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct!")
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect! The time is measured bet

def valueChange(self):
    a = self.horizontalSlider.value()
    b = self.horizontalSlider_2.value()
    diff = abs(a-b)
    self.labelDifference.setText(str(diff) + " msec.")
```

Fig. 18: Vertical bar definition for time extraction

Just as in the previous exercise, when the user is asked questions at the end of the exercise and before proceeding to the conclusions, these had to be coded within a numerical condition (Fig.19)

```
def answerQuestionOne(self):
    x = float(self.lineEditQuestion1.text())
    if (x >= 1 and x <= 2):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! The time is measured between the s
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect! The time is measured between t

def answerQuestionTwo(self):
    x = float(self.lineEditQuestion2.text())
    if (x >= 5 and x <= 6):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! The time is measured between the
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect! The time is measured between t
```

Fig. 19: Code for questions and answers in experiment 2

4.4.4 Experiment 3

Up to now, we have stimulated the sciatic nerve with single shocks. In this experiment we have investigated the effects of repetitive stimulation on muscle contraction. The experiment allows us to give shocks for 2 seconds, and observe the resulting contraction. Each stimulus will be supramaximal, of 4 Volts, and 0.5 msec duration.

We can set the stimulation frequency to any value between 1 and 50 per second. When you press "Go" the computer will plot the result, as a graph of force against time. We must be careful how many times we do this experiment. The muscle will begin to deteriorate with prolonged use, as it is not receiving any blood supply.

When the user selects the frequency to which stimulate the nerve, a graph corresponding to that value is shown. After generating the values by trying out different frequencies and plotting the final graph, the user is asked several questions related to the output obtained. All of them must be replied as they contain important information about the previous laboratory work:

1- When you stimulate the muscle at a low frequency, for example 1/sec, each twitch occurs separately, and the muscle relaxes completely before the next contraction. At higher frequencies, the muscle starts to contract before relaxation is complete. What is the approximate frequency at which the muscle starts to contract before relaxation is complete?

2- When the stimulation frequency is increased, the peak tension becomes greater than in a single twitch. This is called summation. What is the frequency at which summation occurs, approximately?

3- When the stimulation frequency is increased considerably, the individual contractions can no longer be seen. The muscle undergoes a smooth contraction or tetanus. What is the frequency at which tetanus occurs, approximately?

4- What is the maximum force you observed in a tetanus?

5- During a tetanus the nerve is stimulated repetitively. Each stimulus gives an action potential, which is transmitted across the neuromuscular junction to the muscle. Which one of the following statements is true concerning the muscle membrane?

- The muscle membrane remains depolarized throughout the tetanus.
- The muscle membrane fires action potentials at the same frequency as the nerve.
- The muscle membrane fires a single action potential at the start of the tetanus.

After all of these questions are answered, we can conclude that as we have seen in the experiment, the muscle fibres give separate contractions called twitches, when they are stimulated at low frequency (e.g. 5 Hz), and a smooth contraction, called a tetanus, when they are stimulated at a high frequency (e.g. 50 Hz). The ratio of tetanus to peak twitch is about 2:1.

In life muscles contract smoothly, but this is achieved in a more complex way. The muscle fibres are grouped into "motor units" containing 3 to 50 muscle fibres (up to several thousand). Each motor unit is controlled by a single motoneuron. When it fires an action potential, all the muscle fibres in that motor unit contract. The firing of the motoneurons, and the force generated by the muscle, are regulated by feedback from muscle spindles. Fine control of the force of muscle contraction is achieved by varying the number of motor units that are activated (note that they are of different sizes) and by varying the frequency of firing, usually in the range of 5 to 20 Hz. A true tetanus is rare in life [17].

4.4.4.1 Coding files

During this third experiment, a total of five .py files where needed to code all of the windows of this section. For the graph plotting, the frequency election and stimulation buttons had to be connected and coded to generate the corresponding graphs (Fig.20).

```
def valueChanged(self):
    value = self.dial.value()
    self.lcdNumber.display(str(value))


def applyFreq(self):
    value = self.dial.value()
    self.lcdNumber.display(str(value))
    self.changeImage(value)
```

Fig. 20: Code for frequency application buttons

This section of the lab contains a total of five questions, that had to be coded as well under certain conditions, to specify which answers are correct or not (Fig.21).

```
def question1(self):
    inp = int(self.lineEditQuestion1.text())
    if (inp==7):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! Well done!")
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! Go back and make some more observations. Wha

def question2(self):
    inp = int(self.lineEditQuestion2.text())
    if (inp>= 11 and inp<= 13):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! The duration of a twitch is too short to
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! Go back and make some more observations. Wha

def question3(self):
    inp = int(self.lineEditQuestion3.text())
    if (inp>= 30 and inp<= 32):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct!")
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! The correct answer is about 30/sec. During t

def question4(self):
    inp = int(self.lineEditQuestion4.text())
    if (inp==308):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct! Well done!")
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "No! Go back and stimulate

def question5(self):
    if (self.radioButton_2.isChecked()):
        QtWidgets.QMessageBox.about(self, "Correct!", "Correct answer. The muscle m
    else:
        QtWidgets.QMessageBox.about(self, "Incorrect!", "Incorrect!")
```

Fig. 21: Code of questions and answers for experiment 3

4.4.5 Experiment 4

In this last experiment the user will investigate the effect of varying the length of the muscle [18]. It's important to remember that each muscle fibre is made up of many myofilaments which run from one end to the other. There are thousands of repeating units called sarcomeres, which are responsible for the striated structure of the muscle. When the length of the muscle changes, the thick and thin filaments move relative to one another. The experiment starts with the muscle at a minimum length, when the thick filaments are quite close to the Z line. (Fig.22)

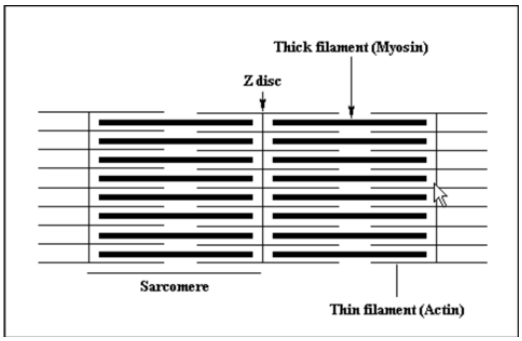

Fig. 22: Representation of thick and thin filaments

The length of the muscle is adjusted by means of a screw thread in the stand to which the muscle and strain gauge are attached. When the thread is turned anti-clockwise through one rotation, the muscle and the muscle bath move down one millimetre, and the muscle is stretched by 1 mm.

In this experiment the muscle is always stimulated tetanically, using supra-maximal stimuli, so it develops the maximum possible force at each resting length. (Fig.23)

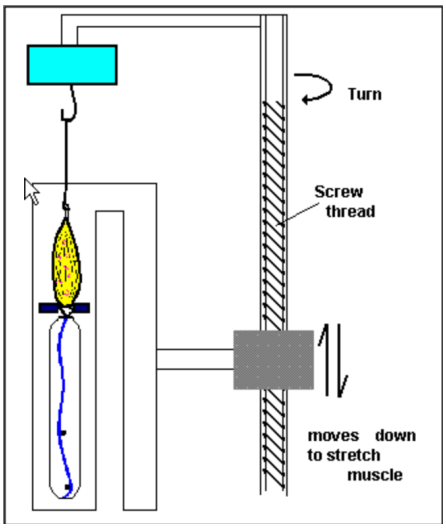

Fig. 23: Set-up of the muscle

In the first part of this experiment, the user can see how the minimum length of the muscle is the length when the thread connecting the muscle to the strain-gauge is just taut. By clicking on 3 consecutive steps, the user can observe graph that shows the record of a tetanus from a muscle at minimum length; then sees the effect of stretching the muscle by 5 mm, then giving it a tetanus; and at last, can observe the effect of stretching the muscle by another 4 mm, then giving it a tetanus. After observing these effects, the user is asked some questions before proceeding to the second part of this experiment:

1- The force generated at minimum muscle length is very small, even in a tetanus. Estimate its value in g weight.

2- Stretching the muscle by 5 mm has increased the force generated in the tetanus. But something else has happened. Did you see what it was?

On the second part of this last exercise, we can set the muscle length to any value between set limits. With that, we are able to explore the effect of changing the length on the resting tension and the tetanic tension. The user can set the muscle length by moving the arrow around the dial (the current length is also displayed in the “Muscle length” box) (Fig.24).

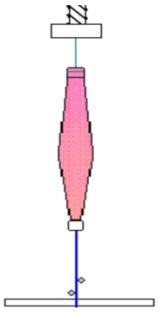

Fig. 24: Muscle set-up

When clicking the "STIMULATE" button, it will then tetanically stimulate the nerve. The resulting resting tension and tetanic tension are displayed in the appropriate boxes in a table. Once the user thinks that have taken enough reading, then can move on to the next page of the software.

At last, the user is asked a last question to make sure that the contents were being followed correctly:

Which of the statements are true?

- When the muscle is stretched, each sarcomere is stretched in proportion.
- The tension generated by cross-bridge cycling is shown by the difference curve on the previous page.
- The muscle has an optimum length for generation of tension.

This experiment was about a complex relationship between the length of a muscle and its tension. First of all muscles resist stretching passively, even when they are not contracting. This resistance is non-linear, and rises steeply at long extensions. Muscles generate force when they are stimulated. This force is greatest when the muscle is at an optimum length, and is smaller at longer or shorter lengths.

4.4.5.1 Coding files

For the last experiment, which is a bit more extent, it was necessary to use a total of seven .py files and therefore seven windows. This section can be divided into two parts. First one is a simple step where the user observes the effect of stretching the muscle when recording a tetanus starting with a muscle at minimum length. For that, three buttons were programmed to show the expected results (Fig.25).

```
def __init__(self):
    super(Experiment4_1_two, self).__init__()
    loadUi("./rsc/ui/experiment4.1_length_tension_two.ui",self)
    self.pushButtonMenu.clicked.connect(self.goToMenu)
    self.pushButtonBack.clicked.connect(self.goBack)
    self.pushButtonNext.clicked.connect(self.goNext)
    self.radioButton1.clicked.connect(self.buttonOne)
    self.radioButton2.clicked.connect(self.buttonTwo)
    self.radioButton3.clicked.connect(self.buttonThree)
    self.progressBar.setValue(int(200/7))
    self.img1 = QPixmap('./rsc/img/exp4/EXP4.1.1.png')
    self.img2 = QPixmap('./rsc/img/exp4/EXP4.1.2.png')
    self.img3 = QPixmap('./rsc/img/exp4/EXP4.1.3.png')

def buttonOne(self):
    self.labelImage.setPixmap(self.img1)

def buttonTwo(self):
    self.labelImage.setPixmap(self.img2)

def buttonThree(self):
    self.labelImage.setPixmap(self.img3)
```

Fig. 25: Code for representing the graphs of a tetanus

This first part was then followed by a short question related to the results presented. As for the second part of this last section, the user gets the opportunity to change the muscle length, and so generate the data for corresponding resting tension and tetanic tension [19]. All of these data are generated within biological ranges and saved in a table, to be afterwards plotted in a graph (Fig.26).

```
        resting_tension = round(random.uniform(50.0,55.0),1)
        tetanic_tension = round(random.uniform(290.5,340.5),1)
    elif (muscle_length >= 30.5):
        resting_tension = round(random.uniform(40.0,45.0),1)
        tetanic_tension = round(random.uniform(330.5,375.5),1)
    elif (muscle_length >= 30):
        resting_tension = round(random.uniform(30.0,35.0),1)
        tetanic_tension = round(random.uniform(330.5,375.5),1)
    elif (muscle_length >= 29.5):
        resting_tension = round(random.uniform(22.0,22.5),1)
        tetanic_tension = round(random.uniform(330.5,375.5),1)
    elif (muscle_length >= 29):
        resting_tension = round(random.uniform(17.0,18.0),1)
        tetanic_tension = round(random.uniform(330.5,375.5),1)
    elif (muscle_length >= 28.5):
        resting_tension = round(random.uniform(13.5,14.5),1)
        tetanic_tension = round(random.uniform(330.5,375.5),1)
    elif (muscle_length >= 28):
        resting_tension = round(random.uniform(11.0,11.5),1)
        tetanic_tension = round(random.uniform(325.5,350.5),1)
    elif (muscle_length >= 27.5):
        resting_tension = round(random.uniform(9.5,10.5),1)
        tetanic_tension = round(random.uniform(290.5,318.5),1)
    elif (muscle_length >= 27):
        resting_tension = round(random.uniform(8.0,8.5),1)
        tetanic_tension = round(random.uniform(180.5,222.5),1)
    elif (muscle_length >= 26.5):
        resting_tension = round(random.uniform(6.5,6.5),1)
        tetanic_tension = round(random.uniform(180.5,222.5),1)
    elif (muscle_length >= 26):
        resting_tension = round(random.uniform(3.0,3.5),1)
        tetanic_tension = round(random.uniform(120.5,144.5),1)
    else:
        resting_tension = 0
        tetanic_tension = round(random.uniform(36.5,39.5),1)
    rowPosition = self.tableWidget.rowCount()
    self.tableWidget.insertRow(rowPosition)
    self.tableWidget.setItem(rowPosition,0,QtWidgets.QTableWidgetItem(""+str(muscle_length)))
    self.tableWidget.setItem(rowPosition,1,QtWidgets.QTableWidgetItem(""+str(resting_tension)))
    self.tableWidget.setItem(rowPosition,2,QtWidgets.QTableWidgetItem(""+str(tetanic_tension)))
```

Fig. 26: Table generation with muscle length, RT and TT data

Just like in the second experiment, a UiComponents function is generated in order to incorporate and plot the graph on a gridLayout.

4.4.6 Conclusions and About

On the last part of the software, conclusions, a general message is shown where the user should have completed each of the four laboratories and worked through the summary questions at the end of each section. Once the student has taken all the notes, and completed the work that they need, the user can exit the laboratory by using the button "Exit". Also, in the "About" section some short information about who designed the software is presented.

5. Results

5.1. Software

The new version of the software, based on original "The Frog Muscle Lab" ended up being a more easier to execute, better appealing and resumed version of the original software. Currently, it can be executed either in Mac or PC without the need of being launched in virtual and downgraded version of Windows as have been done until now. The software itself, is a very primitive and a simple version making it very easy to work with.

5.2. Theory section

As a result of following the theoretical explanations about the frog dissection throughout the pages, the user arrives to the experimental set-up of the laboratory (Fig.27) which made all of the experiments possible. The figure shown is just one of the windows corresponding to the theory and experimental set-up part. Thanks to the correct configuration and calibration of the system, the stimulation of the muscle was possible by varying its' length, frequency or voltage applied.

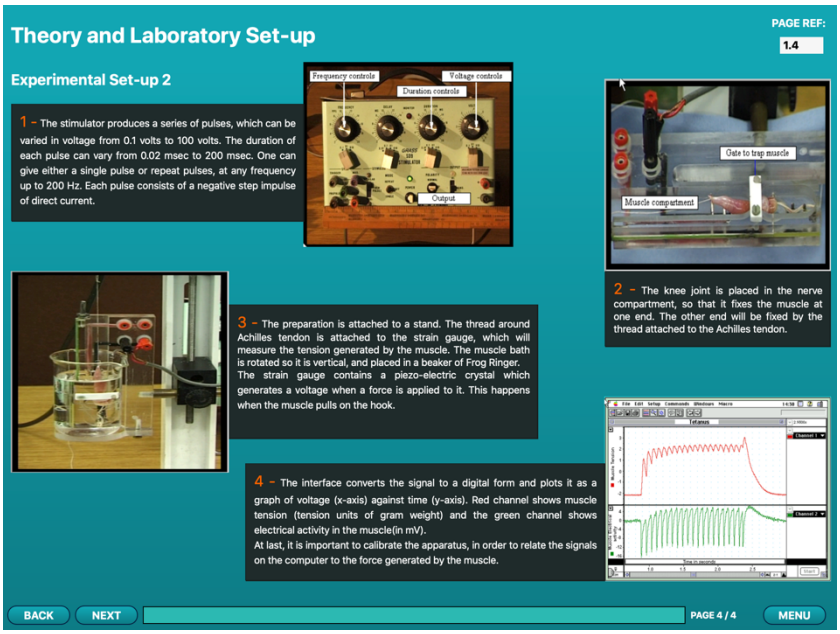

Fig. 27: Initial experimental set-up

5.3. Experiment 1

As previously explained, during the first experiment the user is asked to stimulate the sciatic nerve starting with very low voltage and then increasing it gradually, to observe what is the minimum voltage needed to produce a response[20]. After the first run performing this experiment, we obtain the following graph (Fig.28):

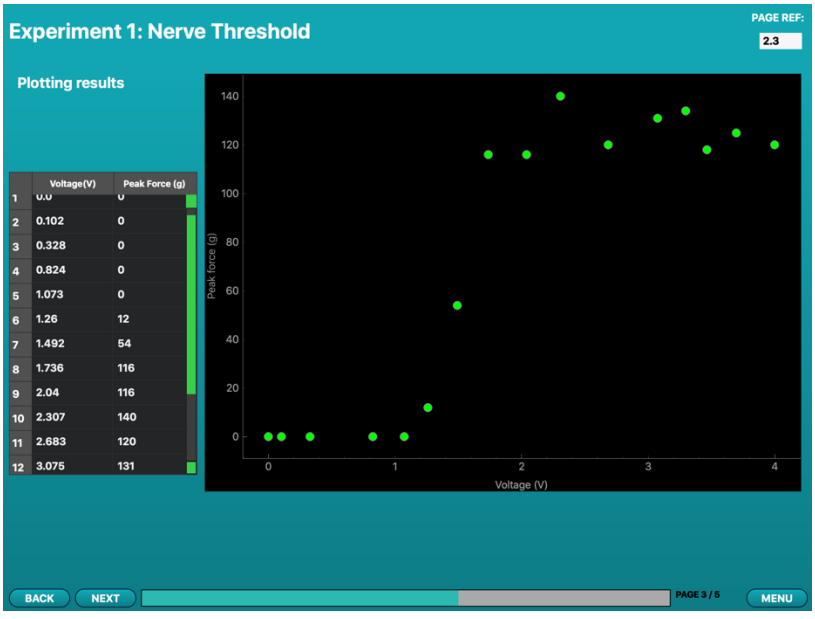

Fig. 28: Plotted results for Experiment 1: Voltage vs Peak Force

It's important to note that the users will never obtain always the same values. That is, in order to simulate biological variability that we observe on daily basis, throughout all

the experiments the values that we obtain are located within a range, so they present biological sense. From these values that the student obtained, must reply the questions that follow next. In case the data obtained is not enough, the user can go back and perform the experiment again, until the data is enough.

5.4. Experiment 2

This section is quite short and the person who is performing it, can finish it and move on to the next experiment quite fast. In this second step of this laboratory, the user can observe and thanks to two sliding bars calculate the time difference between them, which is necessary to observe the time delays between the stimulus, action potential and contraction (Fig.29)

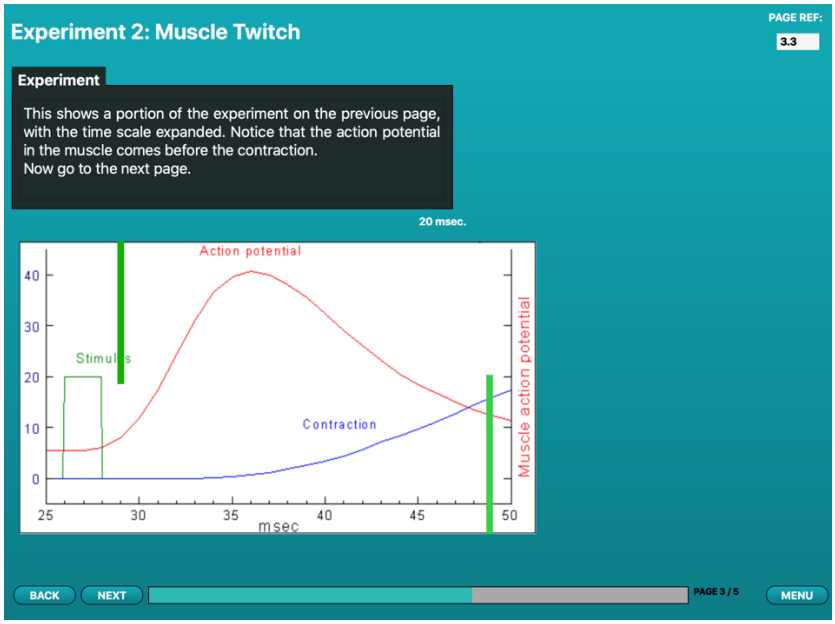

Fig. 29: Plotted results for Experiment 2

Thanks to this graph, the person is able to observe, rationalize and come up with the answers that are asked afterwards in order to be able to proceed to the next Experiment.

5.5. Experiment 3

During this third part, the user can observe very interesting results, by stimulating the nerve with different frequencies between 1 and 50 stimuli per second. Having in mind that each stimulus is supramaximal of 4 Volts and 0.5 msec of duration. When performing a simulation with a frequency of 5, 10, 22 and 50 Hz in the Fig.30 we can observe how the muscle contraction acts accordingly.

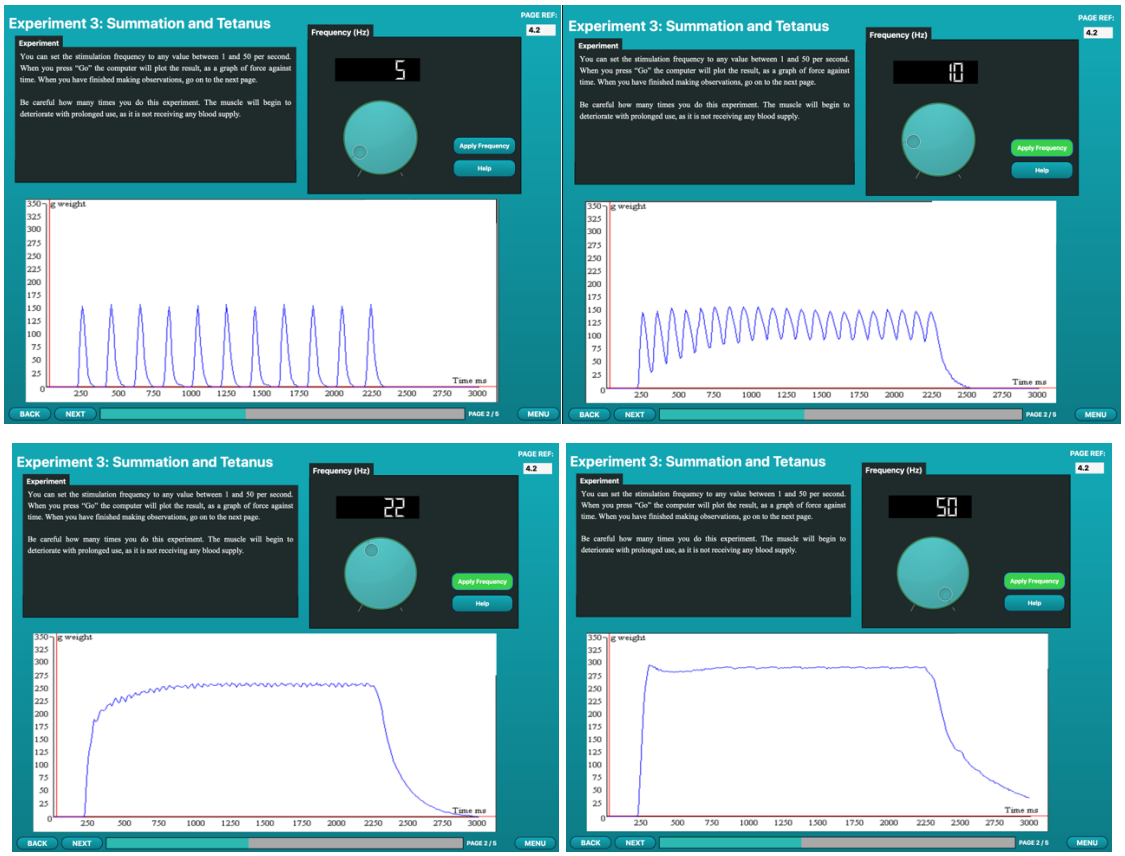

Fig. 30: Result of stimulating the nerve with different frequencies

In the graph we can observe the different muscle contractions in time and how at a low frequency, for example 1/sec, each twitch occurs separately, and the muscle relaxes completely before the next contraction. At higher frequencies, the muscle starts to contract before relaxation is complete [21].

5.6. Experiment 4

During the last experiment, the relationship between the muscle length and respective tension is studied. Each muscle fibre is made up of many myofilaments which run from one end to the other, so when the length of the muscle changes, the thick and thin filaments move relative to one another.

During the first part, the user can observe the graphs of the effects of tetanus on different lengths of muscle (Fig.31)

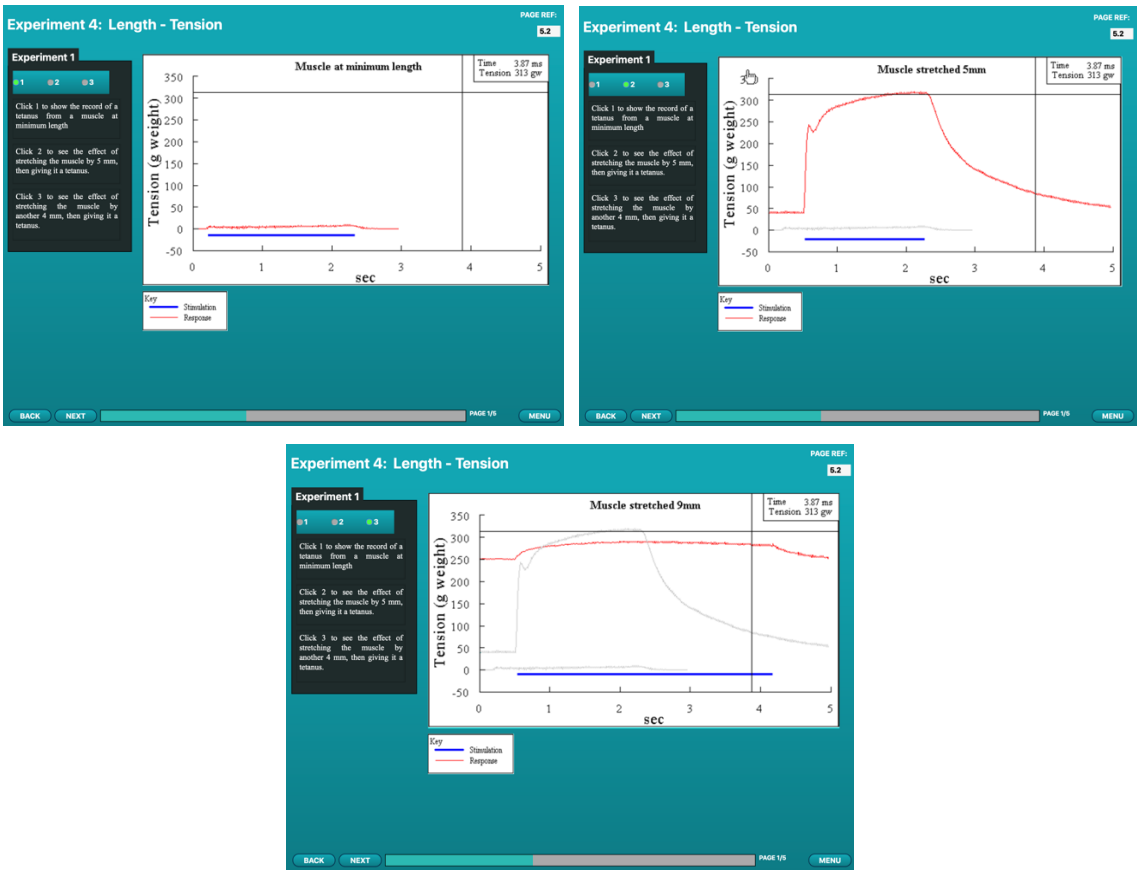

Fig. 31: Effects of tetanus on different lengths of muscle

During the second part of this last experiment, the student can change the length of the muscle to any value between 25,5 and 35 mm. After defining the length, the muscle is stimulated and respective resting tension and tetanic tension are recorded as well. It's important to note that even if the muscle is stimulated twice at the same length, different values shall be obtained, due to biologic variability found in nature. The experimental data obtained by a person are saved in a table (Fig.32), which later on are displayed in a graph (Fig.33)

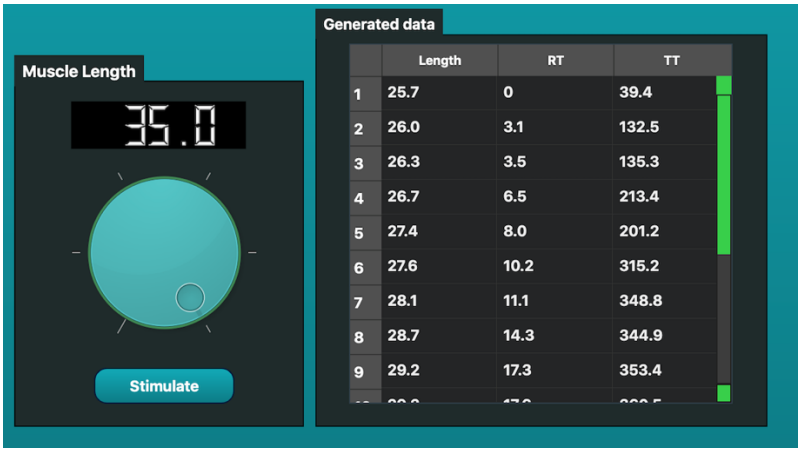

Fig. 32: Data generated and saved in a table

In the table we can observe all of the values that the user generated during this experiment. RT and TT stand for resting tension and tetanic tension respectively. When the program is closed and opened again, all of the values are cleared and reset.

Fig. 33: Plotted results for experiment 4

In this window we can observe the graph with the generated values of RT and TT tensions as well as the difference value between them, which can be seen that at first increases and then decreases as the muscle length increases [22]. From these results we can conclude that the muscle has in fact an optimal length for generation of tension.

6. Conclusions & Future work

From the very beginning, the project proposal was very ambitious that required more time that is assigned for Bachelor Thesis. Nevertheless, all of the parts were covered as referring to the contents and all of the sections were programmed which contain all of the important information. The main objective proposed for this Bachelor Thesis was successfully accomplished. All of the experiments were programmed in a chronological order, meeting the requirements as well as providing the user a very simple and visual platform.

In conclusion, there is still room for improvement and can be considered as future work. Some of the possible enhancements to be proposed are adding more buttons with more explanation or detailed instructions. Also, options such as downloading or printing the graphs obtained could be a good tool for the students who wish to save their results for later analysis.

6. LIST OF FIGURES

BIBLIOGRAPHY

[1] Campbell NJ, Maani CV. Histology, Muscle. [Updated 2021 May 10]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK537195/

[2] Stifani N. Motor neurons and the generation of spinal motor neuron diversity. Front Cell Neurosci. 2014;8:293. Published 2014 Oct 9. doi:10.3389/fncel.2014.00293

[3] KNAPPEIS GG, CARLSEN F. Electron microscopical study of skeletal muscle during isotonic (afterload) and isometric contraction. J Biophys Biochem Cytol. 1956 Mar 25;2(2):201-11. doi: 10.1083/jcb.2.2.201. PMID: 13319381; PMCID: PMC2223965.

[4] Philip M Hopkins, Skeletal muscle physiology, Continuing Education in Anaesthesia Critical Care & Pain, Volume 6, Issue 1, February 2006, Pages 1–6, https://doi.org/10.1093/bjaceaccp/mki062

[5] Skeletal Muscle Fatigue: Cellular Mechanisms, D. G. Allen, G. D. Lamb, H. Westerblad, 01 JAN 2008https://doi.org/10.1152/physrev.00015.2007

[6] Perreault, Eric & Day, Scott & Hulliger, Manuel & Heckman, Cj & Sandercock, Thomas. (2003). Summation of Forces From Multiple Motor Units in the Cat Soleus Muscle. Journal of neurophysiology. 89. 738-44. 10.1152/jn.00168.2002.

[7] What is Python? Executive Summary, 2022; https://www.python.org

[8] Overview about Spyder, 2022; https://www.spyder-ide.org

[9] Learn Python PyQt 2022; https://pythonpyqt.com

[10] About GitHub, 2022; https://github.com/about

[11] QSS Style Sheets Templates, 2022; https://qss-stock.devsecstudio.com

[12] Colorblind-Friendly Color Palettes and ggplot2 Graphic System Extensions for Publication-Quality Scientific Figures. (2021), I: https://doi.org/10.1128/MRA.00871-21

[13] Geoffrey Hammerson, Frank Solís, Roberto Ibáñez, César Jaramillo, Querube Fuenmayor (2004). "Lithobates pipiens". IUCN Red List of Threatened Species. 2004: e.T58695A11814172.

[14] Helmenstine, Anne Marie, Ph.D. "Ringer's Solution Recipe." ThoughtCo, Aug. 27, 2020, thoughtco.com/ringers-solution-recipe-608147.

[15] Layne, Jack R. "Postfreeze Recovery of Gastrocnemius Muscles of Rana Pipiens and R. Sylvatica." Journal of Herpetology 27, no. 4 (1993): 478–80. https://doi.org/10.2307/1564843.

[16] Window, A. L. 1992. Strain gauge technology. London: Elsevier Applied Science.

[17] The American Heritage Medical Dictionary. Boston Massachusetts: Houghton Mifflin Company. 2007. p. 815. ISBN 978-0-618-82435-9. Marked by sustained muscle contractions

[18] Arnold, E. M., Hamner, S. R., Seth, A., Millard, M., & Delp, S. L. (2013). How muscle fiber lengths and velocities affect muscle force generation as humans walk and run at different speeds. The Journal of experimental biology, 216(Pt 11), 2150–2160. https://doi.org/10.1242/jeb.075697

[19] Claflin, D R et al. “The effect of length on the relationship between tension and intracellular [Ca2+] in intact frog skeletal muscle fibres.” The Journal of physiology vol. 508 (Pt 1),Pt 1 (1998): 179-86. doi:10.1111/j.1469-7793.1998.179br.x

[20] Hernández-Ochoa, Erick O, and Martin F Schneider. “Voltage sensing mechanism in skeletal muscle excitation-contraction coupling: coming of age or midlife crisis?.” Skeletal muscle vol. 8,1 22. 19 Jul. 2018, doi:10.1186/s13395-018-0167-9

[21] Interactions of Skeletal Muscles, 2020, LibreTexts Medicine, https://med.libretexts.org/Bookshelves/Anatomy_and_Physiology/Book%3A_Anatomy_and_Physiology_(Boundless)/9%3A_Muscular_System/9.6%3A_Overview_of_the_Muscular_System/9.6A%3A_Interactions_of_Skeletal_Muscles

[22] Frog Sartorius Muscle Length-Tension Curve, 2021, https://www.brown.edu/Departments/Engineering/Courses/En123/MuscleExp/Length_Tension.htm