

UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

**“Development of a Graphical User Interface for
processing and visualization of Brain Computer Interface
experiments“**

Barcelona, June 14th, 2021

Author: Laura Pérez Carasol

Director: Agustín Gutierrez-Galvez

Tutor: Agustín Gutierrez-Galvez

ACKNOWLEDGEMENTS

To my parents, for their unconditional support at all times.

To Miguel and Dani, for their encouragement and for being a true source of inspiration for me.

To my friends, for their great support and advice throughout the development of this project.

To my tutor, Agustín Gutierrez-Galvez, for giving me the opportunity to carry out this project and for his help and support throughout the whole process.

EXECUTIVE SUMMARY

Learning is the process by which new knowledge or skills are acquired and is known to be based on synaptic plasticity and the expansion of the cortical map. However, today it is still difficult to determine the relationship between a specific organizational change at brain level and the learning of a new behaviour. Knowing the functioning of the brain and, in particular, the neural units responsible for the learning process, could make a big difference to those who have suffered a stroke or an amputation and therefore have to relearn the basic locomotor movements.

The project that has been carried out has consisted in the development of a graphical user interface (GUI) that allows the processing and visualization of the data obtained from experiments carried out using Brain Computer Interface (BCI) systems. It has been focused on obtaining the tuning curves, which show the firing rates of the neurons with respect to the angle of perturbation, and the trajectories performed by the subject, in order to subsequently visualize them.

The graphical interface developed consists of a multiwindow application created using MATLAB App Designer based on the data and functions obtained from BCI experiments performed at Carnegie Mellon University. It is mainly aimed at the biomedical sector, although it could be useful in other fields.

Keywords: Graphical User Interface, Brain Computer Interface, tuning curves, firing rates, perturbation, trajectories.

Table of Contents

ACKNOWLEDGEMENTS	1
EXECUTIVE SUMMARY	2
LIST OF ABBREVIATIONS	5
1. INTRODUCTION.....	6
1.1. OBJECTIVES	6
1.2. SCOPE AND SPAN	6
2. BACKGROUND.....	7
2.1. HISTORICAL BACKGROUND	7
2.2. STATE OF THE ART	8
3. MARKET ANALYSIS.....	10
3.1. MARKET OPPORTUNITIES AND TARGET SECTORS	10
3.2. MARKET EVOLUTION.....	11
3.3. FUTURE PERSPECTIVES OF THE MARKET	12
4. CONCEPT ENGINEERING	13
4.1. STUDY OF SOLUTIONS	13
4.2. PROPOSED SOLUTION	15
4.3. ALTERNATIVE SOLUTIONS.....	16
5. DETAIL ENGINEERING	17
5.1. EXPERIMENTS OVERVIEW	17
5.2. GRAPHIC INTERFACE DEVELOPMENT	20
5.2.1. <i>Structure of the experiments data</i>	20
5.2.2. <i>Basic functions</i>	21
5.2.3. <i>Software Basics</i>	23
5.2.4. <i>Design options</i>	28
5.2.5. <i>Graphical User Interface Development</i>	31
6. TECHNICAL VIABILITY	43
6.1. TECHNICAL SPECIFICATIONS AND CHARACTERISTICS OF THE SOFTWARE	43
6.2. STRENGTHS, WEAKNESSES, OPPORTUNITIES AND THREATS.....	43
7. TIMING.....	45
7.1. WORK BREAKDOWN STRUCTURE.....	45
7.2. PERT ANALYSIS.....	46
7.3. GANTT DIAGRAM	47
8. CONCLUSIONS AND FUTURE TRENDS	48
9. BIBLIOGRAPHY.....	50
ANNEX I. CODE USED FOR THE DEVELOPMENT OF THE MAIN WINDOW.....	53
POPERTIES.....	53
STARTUP FUNCTION.....	53

CALLBACK FUNCTIONS	53
COMPONENT INITIALIZATION	54
ANNEX II. CODE USED FOR THE DEVELOPMENT OF THE TUNING CURVES WINDOW	55
PROPERTIES.....	55
METHODS	56
STARTUP FUNCTION	60
CALLBACK FUNCTION	61
COMPONENT INITIALIZATION	80
ANNEX III. CODE USED FOR THE DEVELOPMENT OF THE TRAJECTORIES WINDOW	83
PROPERTIES.....	83
STARTUP FUNCTION	83
CALLBACK FUNCTIONS	84
COMPONENT INITIALIZATION	89
ANNEX IV. CODE USED FOR THE DEVELOPMENT OF THE CONTROL RESULTS WINDOW	92
PROPERTIES.....	92
STARTUP FUNCTION	92
CALLBACK FUNCTIONS	92
COMPONENT INITIALIZATION	92
ANNEX V. CODE USED FOR THE DEVELOPMENT OF THE PERTURBATION RESULTS WINDOW	93
PROPERTIES.....	93
STARTUP FUNCTION	93
CALLBACK FUNCTIONS	93
COMPONENT INITIALIZATION	93
ANNEX VI. CODE USED FOR THE DEVELOPMENT OF THE WASHOUT RESULTS WINDOW	94
PROPERTIES.....	94
STARTUP FUNCTION	94
CALLBACK FUNCTIONS	94
COMPONENT INITIALIZATION	94
ANNEX VII. LIST OF FIGURES AND TABLES.....	95
LIST OF FIGURES	95
LIST OF TABLES.....	95

LIST OF ABBREVIATIONS

- BCI: Brain Computer Interface
- GUI: Graphical User Interface
- MATLAB: MATrix LABoratory
- OMP: Outside-Manifold Perturbations
- PD: Pushing Direction
- WMP: Within-Manifold Perturbation

1. INTRODUCTION

1.1. Objectives

The purpose of this project is to develop a graphical user interface (GUI) for the processing and visualization of data obtained from Brain Computer Interface (BCI) experiments. The data used for the development of this app was obtained from a series of experiments with monkeys using BCI at Carnegie Mellon University. If the creation of this app is successful, it will allow the user to obtain and visualize the tuning curves and trajectories of a particular subject, which will be very useful in determining the evolution of the learning process.

In order to achieve the established objective, a series of secondary goals have been defined that make up the strategic plan to ensure the development of the interface:

- Conduct a study on the experiments on which this project is based and determine the utilities and conclusions drawn.
- Obtain and execute the functions generated during the experiments to visualize the tuning curves and trajectories for a clear understanding of their functioning and to be able to determine the possible changes to be made when incorporating them into the interface.
- Propose and choose the design of the interface to be implemented.
- Develop the GUI using the data and functions obtained from the experiments.

In addition, another of the main motivations for the development of this project is the prospect of, in the future, being able to use this interface with data acquired from humans, allowing further progress in the field of BCI, which is expected to be the future of medical prostheses and neurorehabilitation.

1.2. Scope and Span

The aim of this document is to carry out an explanation of the motivations, the methodology used, and the results obtained in the final degree project of the Biomedical Engineering Bachelor's Degree at the University of Barcelona.

This project is related to the field of Brain Computer Interface systems and has been carried out during the course 2020-2021 for a total of approximately 300 hours, as stipulated in the study plan of this degree. It has been carried out entirely telematically due to the COVID-19 pandemic.

With this work it is intended to provide users related to this field with a graphical user interface that allows to process the data obtained from BCI experiments and to visualize the tuning curves of each neural unit studied, together with the trajectories performed by the subject during the experiments. This is expected to help researchers determine how neural electrical activity is affected in the process of learning a skill and which neurons are involved in this process.

2. BACKGROUND

2.1. Historical background

Before the development of BCI systems, certain innovations were made in the field of electrical activity in the human brain. A scientist that is important to highlight is Hans Berger [1] who was the creator of electroencephalography, which allowed scientists to study electricity in the brain as well as the final development of the electroencephalogram (EEG). Furthermore, since this project focuses on the processing of data obtained in animal experiments, it is worth mentioning the discovery of brain electrical signals in animals made by Richard Canton in 1875.

The first major research on BCI was carried out in the 1970s by the Defense Advanced Research Projects Agency (DARPA), which is an agency of the U.S. Department of Defense responsible for the development of new technologies for military purposes. But it was not until 1973 that the term "brain-computer interface" was first used, when a specialized computer scientist named Jacques J. Vidal [2] published a paper introducing the concept of this technology based on signals obtained by electroencephalography in which several electrodes are placed on the patient's skull and data on the electrical activity of the brain is collected. Since then, scientists have attempted to understand brain waves in order to control external devices.

One of the landmark events in mapping the electrical activity of the human brain was the first brain implant performed in 1998 by researcher Philip Kennedy on a human subject that allowed high-quality signals to be captured by means of a wireless dielectrode.

Later, in 1999, Birbaumer and his research team developed a device based on a BCI system that allowed communication to patients affected by total paralysis. For this purpose, slow cortical potentials obtained by an EEG were used to control an

electronic spelling device [3]. A year later, the research group led by Miguel Nicolelis [4], professor at Duke University (North Carolina), achieved prolonged control of a robotic arm to reach for food or a cursor using real-time transformations of neural signals derived from multiple cortical areas of owl monkeys. This research used an open-loop BCI in which the primate could not see the moving arm and therefore did not receive feedback.

One example of the application of BCI technology in a human subject that is worth mentioning is the case of Matthew Nagle. He was the first patient with tetraplegia who was able to control a robotic hand using a BCI system to regain functionality lost due to paralysis. This 2005 clinical trial was part of a 9-month human trial to test the efficacy of a chip implant called BrainGate developed by Cyberkinetics. This implant, consisting of 96 electrodes, was implanted in the right precentral gyrus, which corresponds to the part of the brain that controls arm movement. This trial was a great success and allowed Matthew Nagle to use a computer cursor or a remote control for the first time [5]. It is also important to note that, in the same year, the Blue Brain project, promoted by IBM and the École Polytechnique Fédérale de Lausanne in Switzerland, was launched. This project was created to build a computer model (a virtual brain) in which the structure and functioning of a human brain could be contained, with the main objective of being able to load the knowledge and sensations of a human brain into a machine [6].

In the last 10 years, there have been major advances in this field, such as brain-to-brain communication, called BrainNet, which consists of a direct non-invasive brain-to-brain interface for several people, by combining electroencephalography to record brain signals and transcranial magnetic stimulation to deliver the emitted information noninvasively [7].

2.2. State of the art

Currently, when developing the software for BCI systems, a combination of MATLAB and Simulink is mostly used. These platforms must offer blocks where the data to be studied can be imported, and presentation modules to show the results obtained after extracting the most relevant information. Most of these software packages are usually developed by the laboratories themselves using a wide range of programming languages and tools that are not usually available to the public. However, there are some companies such as the one launched by entrepreneurs from the University of

Zaragoza, called BitBrain, that offer, among other products, software solutions for neurotechnology.

In addition, certain platforms are currently available for developing and implementing BCI systems [10]. The BCI2000 consists of a general-purpose software platform for BCI research, which, although programmed with the C++ language, allows inline signal processing code to be written in MATLAB and includes a full Python compatibility layer. It can incorporate one or a combination of brain signals, signal processing methods, external devices and operating protocols. This system works well in online operations and meets the real-time requirements of BCI systems. This platform facilitates the implementation of different BCI systems, as it reduces manpower and costs. In addition to brain signals, the design of this system also allows using inputs from other devices (cursor, keyboards...) [11]. Besides this platform, two others have the necessary functionalities for real-time BCI designs: BCI ++ and BioSig.

Furthermore, a free and open source software platform called OpenViBE, available for Windows and Linux operating systems, allows users to design, test and use brain-computer interfaces. Thanks to its simple user interface, this platform can be used by people without programming skills and can be operated with different acquisition machines (EEG or MEG). An outstanding feature of OpenViBE is the facility with which it can be integrated with other applications such as virtual reality [12].

Finally, there are several more platforms related to the field of BCI such as TOBI which is a set of cross-platform interfaces which connect parts of different BCI systems, BCILAB which consists of an open source MATLAB-based toolbox for advanced research in this area, and xBCI, a generic platform to develop online brain-computer interfaces easily and quickly thanks to its easy-to-use system development tools, among others.

3. MARKET ANALYSIS

3.1. Market opportunities and target sectors

BCIs are gradually being introduced in the vast majority of market sectors [13]. This technology has caused a breakthrough in the field of neuroergonomics and the intelligent environment, since it has allowed, for example, the development of a cognitive control system called the Environmental Self-Tuning Control System based on the Smart Computer brain interface (BSLEACS). This system adapts the environment components according to the user's mental state. In this sector, it has been seen that operating rooms would be great candidates for intelligent applications based on BCI, since it would mean great benefits for the surgeon [14].

Another sector that has benefited from this technology is transport, since intelligent driving systems have been developed to detect the driver's cognitive state. This allows, through the use of signals obtained from an electrocardiogram (ECG) and an electroencephalogram (EEG), to regulate the speed of the vehicle according to the driver's level of concentration and stress [15].

In the field of advertising, politics, and education, BCI systems are also beginning to be used. Besides, this technology is also being applied in the video game sector, which allows combining the characteristics of existing games with the innovation of mind control. An example of a video game that uses this technology is BrainArena, in which players can play a soccer game simply imagining movements with the left or right hand. In addition, there are games more focused on emotional control such as Brainball, which aims to reduce the user's stress by making him/her move a ball. Since in this game the player who is less stressed wins, it is a good way for the user to learn to control stress through an entertaining activity.

But the applications of BCI technology in the medical field [16] should be highlighted above all. In the prevention sector, BCI systems can be used to predict dizziness in drivers and thus develop a system to monitor and alert the driver's condition by means of EEG power indicators and thus prevent traffic accidents caused by loss of attention due to dizziness [17]. In addition, it could be applied to sick elderly people living alone to monitor their health. In the detection and diagnostic phase, several applications have been studied that would allow the detection of brain tumors, seizure disorders and brain inflammation [18]. Finally, BCI technology can also be applied to the necessary rehabilitation after suffering a stroke or in the case of paralysis, as studies

have been conducted in which the use of prostheses, called neuroprosthetic devices, can allow the full recovery of limb functionality [19]. Moreover, this technology is considered to be the future of medical prostheses.

3.2. Market evolution

The market to which BCI technology is directed has been evolving in recent years [20]. In the beginning, this interface emerged primarily as a binding technology between the electrical signals produced by the brain and the device used to detect the resulting encephalogram. It was not until a decade later, in 1988, that the brain-machine interface began to be used for non-medical purposes. Farwell and Donchin were the first to introduce the paradigm called “P300-speller”, known worldwide today. These scientists developed a BCI system that allowed the use of event-related potentials (ERP), which are deviations from an EEG that arise due to the subject's response to a specific event or stimulus.

At the end of the last century, new brain-machine interface systems based on visually evoked steady state potentials (SSVEP) were introduced [21]. These potentials consist of oscillatory electroencephalography activities whose frequency is synchronized with that of a blinking visual stimulus. This type of system allows associating the blinking frequency corresponding to the specific stimulus applied to the subject with specific BCI commands, and was implemented in a flight simulator in which the left and right movement of the aircraft was controlled using two flashing lights placed on the left and right side of the cabin. On the other hand, in 1996 a more specific and advanced machine learning was established for the BCI, in order to classify the signals obtained using an electroencephalogram in a more robust way. This was initially done using support vector machines or neural classifiers, although different algorithms were later implemented. One of the most widely used standards for machine learning in BCI is the spatial filtering algorithm for common spatial patterns (CSP) proposed in 2000 by Ramoser.

Regarding the start of invasive animal BCI investigations, the first ones were carried out between 1999 and 2001 by the research group led by Nicolelis. Rats were used in the first tests, although later they opted for primates. In these investigations it was found that the animals used were able to control robotic arms using only the signals emitted by neurons located in the motor cortex. These signals were obtained by implanting electrodes into the subjects' brains.

On the other hand, if the sectors in which this technology has been introduced are considered, it should be noted that initially it was only used in the medical field. Although BCI has maintained most of its implementation for the healthcare sector, whether for disease prevention, tumor detection and rehabilitation after a stroke, it has also been introduced in sectors such as neuroergonomics, intelligent environments, transportation, advertising, politics, education, etc. The use of this technology has not had a sudden entry into these latter sectors but has been increasing its influence as technology advances.

3.3. Future perspectives of the market

BCI is a breakthrough technology that is advancing rapidly due to the fact that its applications in the vast majority of sectors are highly desired by businesses and governments. Moreover, thanks to the speed of technological advances, the possibility of increasing the scope of action and the number of applications of BCI is growing [22].

A few factors should be highlighted for their influence on the progress of this technology. First, the fact that the number of incidents of mental disorders affecting the motor parts of the body is increasing, leads to greater social pressure for the development of technologies to restore mobility to those who suffer from it. According to the U.S. National Institute of Mental Health, one out of four adults in the U.S. suffers from mental disorders each year and 6% of the population suffers from severe disabilities, and these numbers are expected to increase over the years. Another factor that may greatly influence the progress of brain-computer interfaces is the advance of flexible circuit technology, which allows the components of an electrical circuit to be miniaturized. This technology allows the use of pressure sensors in blood vessels, in the heart or even the implantation of chips directly in the brain, which increases the feasibility of wearable BCI technology, since only one device would be needed to capture the data obtained through sensors or chips. Finally, more and more opportunities are emerging, especially in healthcare. It is expected that healthcare infrastructures in developing economies will grow, together with their governments contributing funds to improve the welfare of the population. As a result, an increase in subsidies for the development of new technologies and new advances in BCI technology is expected.

One fact provided by the World Health Organization (WHO) worth noting is that by 2030, around 82 million people will suffer from dementia and the number of cases of

Alzheimer's, Parkinson's and epilepsy will increase dramatically. This will lead to an increase in demand for technologies, such as BCI, that improve the quality of life for people suffering from these disorders and other physical disabilities.

As for the prospects for new applications and new products related to BCI technology, it has been predicted that the first computers capable of simulating a human brain will emerge between now and 2045. This will enable the transplantation of human brains into robots and the complete understanding of the functioning of the human brain. In the very long term, assumptions have been made about the advancement of this technology, including the possibility of dream visualization and interpersonal communication using people's thoughts wirelessly. In addition, it is said that by the year 2090, scientists will be able to transfer the thoughts and knowledge of a deceased person to a computer, which would mean the immortality of the human brain.

4. CONCEPT ENGINEERING

4.1. Study of solutions

Several options are available for the development of the visualization and data processing tool regarding the software selection. In this section the different options available on the market in terms of programming languages will be outlined, and the advantages and drawbacks considered for the choice of the most optimal programming language will be mentioned.

In the table below, the programming languages that have been discarded after the first exhaustive study of the different programming languages that are used today by programmers can be seen. Despite the fact that all seven of these languages have significant positive points, they are characterized by having a steep learning curve, a non-intuitive code readability, such as PHP, Perl, Java, C++ and C#, a slow code execution and data processing, such as JavaScript and Java, or by being quite inflexible, such as the Swift language [23].

Programming language	Positive points	Negative points
PHP	Easy to use Open source	Need for a web server Generic HTML knowledge needed Most used for web development
Perl	Simple Built-in interpreter	Slow Hard to read code Difficult troubleshooting
Java	Multi-platform Free distribution Complete	Slow execution Complicated learning
C++	Didactic Programming with multiple styles	Complex for programming databases Use of libraries is complicated
C#	Powerful and flexible	Difficult to do portability Complicated learning Lack of documentation for the tool
JavaScript	Easy to use Complete	Few resources Vulnerable Need to download the code before processing
Swift	Easy learning Very secure code	Imposition of a large number of rules to program Recent creation

Table 1. Software comparison

As for the programming languages chosen as possible candidates for the development of this project, Python, R and MATLAB stand out. The advantages and disadvantages of each of them can be seen below.

Programming language	Positive points	Negative points
Python	Simple Quick Flexible Very portable Required libraries installed inside the provided interpreter Widespread community Open Source	Relatively complicated learning Bad documentation
R	Quick calculation Open Source High speed in data processing Allows large volumes of data Multi-platform Simple error correction	Bad documentation Erratic tool for machine learning projects Slow
MATLAB	Fast execution High accuracy Extensive mathematical support Extensive support of already developed functions Integration with Hardware devices Widespread community	"Dark" memory management Eventual speed problems

Table 2. Final software comparison

As it can be seen, these languages have a large number of advantages, among which it is worth highlighting the rapid execution of the developed code, their flexibility, and the fact that all of them are widely used by a large community, which facilitates the solution of errors.

4.2. Proposed solution

As mentioned above, the development of the GUI generated in this project has been possible thanks to the data and functions obtained from BCI experiments carried out by Carnegie Mellon University (USA). Since the different functions generated were developed using the MATLAB programming language, the choice of the language to be used was determined mainly by external factors.

Regardless of this issue, the choice between Python, R and MATLAB would have resulted in this programming language. One of the reasons for this is that the MATLAB programming language is highly accurate and fast when it comes to processing data and performing mathematical operations. Moreover, it stands out for its broad mathematical support and for having a large number of functions already developed, which greatly facilitates code development. An outstanding feature of MATLAB is that it has a wide community, which allows a faster resolution of errors that may occur during the development of the software. This will also facilitate the resolution of any doubts that may arise during the process and allow the development of an optimized, error-free software.

As already mentioned in the main objectives of the work, the purpose of this project is to develop a GUI for the visualization and processing of the data obtained through the experiments performed with BCI systems. For this purpose, the fact that the MATLAB programming language allows the elaboration of a large number of graphs and images and that these can be displayed on any graphical output device, makes it an exceptional tool for visualization of technical information. In addition, the MATLAB software provides an interactive development environment in which applications can be designed while programming the behavior of each of its elements.

For the development of this project, two computers with Windows and iOS operating systems will be used. This will not pose any problem when programming in MATLAB, since this numerical calculation system is independent of the platform on which it is used, which means that code can be developed using this programming language on

iOS, Linux, Windows and UNIX, and can be run on any of the operating systems. In addition, the information files that are written can be read on any platform. Thanks to MATLAB's platform independence, information and code can be exchanged between different computers regardless of their operating system.

The negative point that could lead to the rejection of this option would be its cost, since it is five to ten times more expensive than a C or FORTRAN compiler. This has not seriously affected the choice of the programming language to be used to carry out this project, since this cost is covered by the University of Barcelona.

4.3. Alternative solutions

This section will show the alternative solutions that have been considered in case the project could not be developed using the MATLAB platform.

The first option that has been studied as a programming language alternative to carry out this project has been Python. This decision was based on the study and balance of the advantages and disadvantages that it could provide. One of the positive points to emphasize is that this computer language is flexible when programming and stands out for its speed in code execution. In addition, it provides the user with a large number of libraries already installed in the provided interpreter, avoiding the time-consuming task of searching, downloading and importing libraries available on the Internet. On the other hand, it should be noted that Python also has a large community of users that would facilitate the development of the code and would allow most of the errors or bugs produced when programming the software to be resolved quickly and easily. The reason why this programming language was discarded as the first option was because it is complicated and slow to learn, which would make it necessary to extend the deadline established for the development of this project.

A second alternative to be used in the event that the first two options were not possible would be the R programming language. Like the other main options proposed, this computer language is characterized by its calculation and high data processing speed. In addition, R allows an easy correction of the errors and is able to support large volumes of data, which is essential for the development of this project since the software to be developed will have to be able to process a large amount of information. On the other hand, the disadvantage that has made R the third option and not the first

one is its slowness when executing the programs, which is not desirable if, in addition, the code is extensive and there is not much time to develop it and check its efficiency.

In the case that none of these options were possible, a more exhaustive study of the other proposed options would be made in order to choose the most suitable one for this project.

5. DETAIL ENGINEERING

5.1. Experiments overview

It is known that learning is based on synaptic plasticity and the expansion of the cortical map, in other words, it is related to organizational changes that occur in the brain. However, it is still challenging to determine a relationship between a specific change and the learning of a new behaviour. The experiments on which this project is based, carried out by Carnegie Mellon University, attempt to demonstrate that new neural networks can be created through learning and enable new behaviours or skills.

In order to demonstrate this, it is necessary to first determine which neurons are responsible for the behavioral changes that are learned. The use of a BCI system facilitates the task, since it allows to detect and establish the relationship between neuronal activity and the behavior shown by the subjects, which makes it a very useful when studying the learning process.

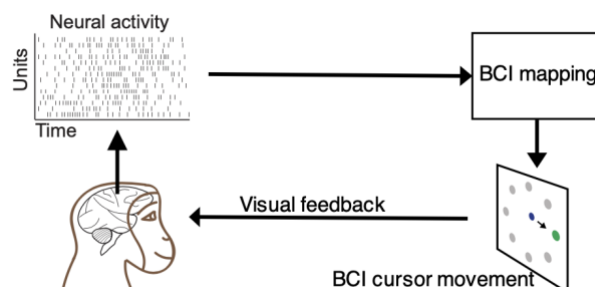


Figure 1. Schematic of a BCI system¹

The main hypothesis established when carrying out these experiments was that the acquisition of new skills is due to the formation of new patterns of neural activity. To

¹ Image extracted from the article referenced in [24]

affirm this theory, it was necessary to motivate the formation of new neural patterns to detect whether these patterns are indeed produced and, if so, to determine whether they are directly related to learning or not. To carry out these three steps, a paradigm using BCI was used, as previously mentioned, in which a monkey had to move a cursor from the center of the screen to one of the possible targets located in the periphery, which were shown one by one. It should be noted that by using this paradigm, it was established that the cursor movement is solely due to the neural activity in the studied population, which implies that any mismatch between the desired cursor movement and the decoded cursor movement can only be corrected by altering the activity of these recorded neurons. The information from these trials was collected using a multi-electrode array, which was chronically implanted in the primary motor cortex of the subject, specifically in the arm region.

The daily learning capacity has been shown not to be unlimited, but to be bounded by the activity structure of the neural population. The "intrinsic manifold" was therefore established as how neurons covary naturally, intuitively. Because of this, two types of BCI mappings can be created depending on how consistent it is with the "intrinsic manifold". This is important when determining whether new neural networks are created or not, since one-day learning of a within-manifold perturbation (WMP) is easily achievable since in the case of this type of perturbation only the reassociation of existing neural activity patterns with different movements is necessary and, therefore, no new patterns are created. In the case of outside-manifold perturbations (OMPs), the opposite occurs since they are inconsistent with the "intrinsic manifold" of the individual and, therefore, they cannot be learned in a single day since new neural patterns must be formed.

Experiments were conducted on two monkeys. For the development of the graphical interface created for this project, data from only one of the monkeys, named Arthur, were used. Before starting the experiments, the "intuitive neural repertoire" was defined based on the patterns of neural activity prior to learning. Subsequently, the patterns that form the "intuitive neural repertoire" of the neural units were projected onto the OMP map as 2D cursor velocities, which defined the speed limit. After learning, it was observed that if the monkey managed to generate velocities higher than the established speed limit, it meant that these had been generated due to the creation of new patterns of neural activity, since these must be beyond this repertoire for the velocity to be outside the limit. As a result, the percentage of neural activity was seen to increase with time, indicating that the brain is capable of generating new patterns, although learning is not imminent. In addition, the speed component in the

target direction was observed to determine the level of progress, as high level of progress means faster and straighter course movements.[24]

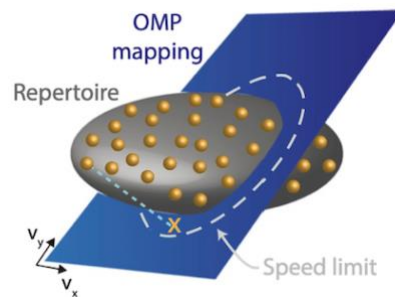


Figure 2. Intuitive neural repertoire obtained while using an intuitive mapping.²

Using the electrodes implanted in the subjects, the firing rates produced by the neurons to be studied were collected, which allowed, through the use of a series of equations, the determination of the decoding preferred direction (dPD) of the neurons.

The BCI paradigm used in this study consisted of a series of experiments divided into 3 phases. The first one is the control phase, during which an intuitive BCI mapping between neural activity and cursor movement was used for several days in order to establish baseline performance metrics. To establish this mapping, a fitting with a cosine-tuning function centered on the PD of each neuron was performed and, subsequently, the velocities were estimated using the population vector (PV) algorithm as the sum of the PD of each unit weighted by its normalized firing rate. [25] This was followed by the perturbation phase, in which the intuitive mapping was changed to a perturbed BCI mapping, where the relationship between neural activity and cursor movement was modified to motivate learning, for a few days. The mapping modification was performed by rotating the dPDs of specific neurons at a given angle. This type of mapping is called Credit Assignment Rotation Perturbation (CARP) and causes the cursor to move at an angle relative to the desired direction or to move more slowly. Thanks to this type of mapping it was possible to see if the subjects were able to use the error signals and correct their movements, identifying and modifying the set of neurons causing these errors, being the ones that have been perturbed. The last phase is called washout, where the original dPDs of all neurons are restored to revert to the intuitive mapping used in the first phase for several days or weeks. In this latter phase it was seen, especially in the first weeks, that the errors made by the subject were of similar magnitude to those occurring in the perturbation phase but in the opposite direction, and it could be observed that the unlearning process is slower than the learning process. To consider that a trial had been successful, it was determined

² Image extracted from the article referenced in [25]

that the monkey had to manage to move the cursor to the target in less than 3 s. It was found that in the case of the control phase, the success rate was high, which makes sense because an intuitive mapping was being used and therefore no learning process had to occur [26].

Therefore, it could be concluded that fast learning can be achieved by reassociating existing neural activity patterns, whereas slow learning usually involves the creation of new patterns which would involve changes in the tuning curves of specific neurons.

As results of these experiments, the trajectories performed by the subjects, which allows to see the evolution of their learning progress, and also the tuning curves of each neural unit studied, which consist of plots where their firing rates are represented with respect to the angle of perturbation, were obtained. The changes in the tuning curves of the neurons is what causes the improvements in learning and, therefore, the comparison of the tuning curves in the different phases is very useful to see if these have undergone a great change or if they have remained the same, which would indicate that this neural unit in question does not intervene in the learning of the movement towards that target.

The importance of being able to visualize the evolution of the trajectories and tuning curves is the reason why it was decided to develop the graphical interface created in this project, aiming to facilitate the user the processing and visualization of the data obtained in the experiments in an effortless and visual way.

5.2. Graphic interface development

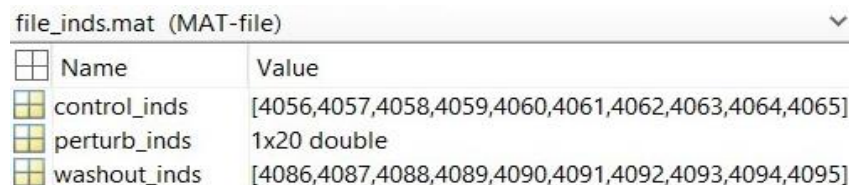
Prior to the development of the graphical user interface, the structure of the data and functions generated during the experiments used to obtain the tuning curves and trajectories were analyzed and studied.

5.2.1. Structure of the experiments data

The results obtained from the experiments carried out in the United States at Carnegie Mellon University are gathered in a folder with the name of the monkey, Arthur. Within this folder there are different subfolders, each of which corresponds to a specific day on which a series of trials were carried out. Each file corresponding to a trial is defined

by a series of four or five identifying numbers, and all file names have a common structure, *Arthur.BC.#####.CenterOut.mat*, which is important to take into account when importing the data into the graphical interface.

In addition, inside each folder there is a file called *file_inds.mat* which contains the identification number of the files corresponding to the three phases of the experiments: control, perturbation and washout. This will allow access to the specific files of a given phase, which will be very useful when visualizing the data.



Name	Value
control_inds	[4056,4057,4058,4059,4060,4061,4062,4063,4064,4065]
perturb_inds	1x20 double
washout_inds	[4086,4087,4088,4089,4090,4091,4092,4093,4094,4095]

Figure 3. Contents of a *file_inds.mat* file

As for the structure of each of the files, these are composed of 9 structures in which all the information collected during the experiments is stored. These include the “header” structure, where the basic information of the specific experiment is included (date, time, monkey name, etc), the “spikes” variable, where all the spikes produced during the experiment by each of the target cells studied are collected, and the “em_feedback” structure, which contains the cursor position at each moment and the location of the targets used. In addition, within the files there are also three structures that correspond to the information of the successful, fail and catch trials.

The data to be loaded in the graphical interface developed in this project must have the same structure as the one mentioned in this section, otherwise it will not be possible to correctly read the information needed to visualize the tuning curves and trajectories.

5.2.2. Basic functions

As for the functions used to develop this project, the first one studied was the *Visualize_tuning_curves* function. In it, the days and the target cell of interest, and the files to be studied are established, and the corresponding file *file_inds.mat* is loaded, where the identification number of the files corresponding to each of the three phases of the experiments is indicated. Subsequently, the file of interest is imported.

Once the file is loaded, a subfunction called *get_seq_rate_info_short_win* is executed. This subfunction provides the rates, which correspond to the frequency of response of

a neuron, for different directions, corresponding to the targets used. The input that this function receives is the response, the spikes that each of the neurons have in a certain interval of time and for several experiments. In addition, two other inputs required by this function are the range mode and the regress mode. The first one refers to the type of window to be used when counting the firing rates, for which the "half_rate" option was chosen. As for the regress mode, this corresponds to the regression mode being used. In this case it was decided to use a regression on the number of spikes/bandwidth, i.e. the number of spikes in a certain time window is counted and divided by the time of the window, which corresponds to the option "rates". Finally, the size of the spike window must also be specified, which has been set to 0.3. Short_win refers to the fact that it uses a short time interval to count the spikes. To obtain the rates, this subfunction detects how many activations, which correspond to a higher frequency of electrical activity, there have been in a certain period of time. For each target position, the activation of that neuron, which is proportional to the number of spikes in a given time interval, is sought.

Therefore, by using this subfunction it is possible to obtain a series of outputs including the "info" variable where the basic information about the experiments is collected such as the obtained rates, among other things, and the "cellnames" variable where the target cells available for the day, the phase and the set files are included.

Once these variables are obtained, the index of the position of the desired target cell in the list established in the variable "cellnames" is determined. Finally, the positions of the targets are ordered from smallest to largest angle to facilitate the visualization, and only the rates corresponding to the desired target cell are selected, which is done by extracting exclusively the column determined by the index obtained previously.

This whole procedure is carried out for all the files corresponding to the selected days and phases chosen by the user, and the rates obtained for each case are added to a matrix called "all_rates".

Before visualizing the tuning curves, a vector is created in which the average of the spiking rates of each angle is included to obtain a representative value for each case. To obtain the final figure, a graph is created with all the rates obtained as a function of the angles, on which a line graph created from the average values obtained is added.

On the other hand, to visualize the trajectories produced by the monkey during the experiments and thus be able to see the evolution of its learning, two functions developed during the experiments were used.

The first function, *extract_trajectories*, allows, as its name suggests, to extract the trajectories produced by the cursor in a given experiment. The operating mode of this function consists of loading the desired file in order to access and extract the information about the cursor positions throughout the experiments. These positions are located in a nx3 matrix within the "em_feedback" structure since the positions were generated taking into account the 3 coordinate axes x, y, and z as three-dimensional targets were used in some experiments. However, for the development of this project only a 2D plane will be considered. First, the zero positions of the cursor, located at coordinates (0,0,0), are found in the matrix in order to determine the positions corresponding to each trajectory. This is because the range of positions corresponding to a trajectory will go from a zero position to the position before the next zero. Once the cursor positions corresponding to each trajectory have been obtained, using the function *Visualize_trajectories*, they are plotted together with the corresponding targets, whose positions are defined in the "trials" structure within the "TargetPos" variable.

5.2.3. Software Basics

This section will describe the software used for the realization of this project. First, a basic introduction of the programming language used will be given, followed by a description of the environment employed for the development of the graphical interface and the basic components used.

MATLAB

MATLAB³, an abbreviation of MATrix LABoratory, is a programming and numerical computing platform that provides the user with an integrated development environment (IDE). This platform has its own programming language, M, which is a matrix-based language and has the advantage of being a multiplatform program, which means that it is available for Windows, macOS, Unix and GNU/Linux. This will be especially useful because macOS and Windows operating systems will be used for the development of this application.

The following is a brief description of the most valuable functions that have allowed the execution of this project in order to facilitate the understanding of the developed code.

- **Load** (*filename*): This function enables loading the desired file into the workspace.⁴

³ For more information about MATLAB visit <https://es.mathworks.com/discovery/what-is-matlab.html>

⁴ For more information about the Load function visit <https://es.mathworks.com/help/matlab/ref/load.html>

- **Ismember** (*A*, *B*): The output of this function determines whether the element(s) of variable *A* are also present in variable *B*. This function returns an array of logical values: 1 if the values are present in *B*, and 0 if they are not.⁵
- **Length** (*x*): This function can be used to determine the value of the length of a vector or a matrix. In the case of a matrix, the function will return the length of the largest array.⁶
- **Size** (*A*): This function provides as output a vector in which the dimensions of the matrix are found.⁷
- **Isempty** (*A*): By using this function it can be determined whether an array is empty (1) or not (0).⁸
- **Dir** (*path*): This function lists the contents of the folder in question.⁹
- **Strcmp** (*s1*, *s2*): With this function it is possible to compare arrays of strings, character vectors and arrays of cells of character vectors. Its output is of logical type, which means that if all the elements of the two arrays are identical, the function will return a 1, and if not, a 0.¹⁰
- **Strcat** (*s1*, ..., *sN*): This function is used to concatenate strings horizontally.¹¹
- **For**: It enables a group of instructions to be executed a certain number of times. To do this, it is necessary to include an index in which it is indicated how many times the commands inside the loop are to be repeated.¹²
- **If/elseif/else**: Using this type of loop, it can be checked whether a condition is fulfilled or not. If the established condition is met, the instructions inside the if statement will be executed, but if not, the instructions inside the else statement will be the ones executed. In addition, if a second condition is needed, an elseif statement can be added.¹³
- **Try/catch**: The use of these commands has been very useful in this project since it shows if any error has occurred during the execution of a series of functions, which facilitates their localization.¹⁴
- **Switch**: This function evaluates an expression and chooses one of several groups of statements to perform the execution.¹⁵

⁵ For more information about the Ismember function visit <https://es.mathworks.com/help/matlab/ref/double.ismember.html>

⁶ For more information about the Length function visit <https://es.mathworks.com/help/matlab/ref/length.html>

⁷ For more information about the Size function visit <https://es.mathworks.com/help/matlab/ref/size.html>

⁸ For more information about the Isempty function visit <https://es.mathworks.com/help/matlab/ref/isempty.html>

⁹ For more information about the Dir function visit <https://es.mathworks.com/help/matlab/ref/dir.html>

¹⁰ For more information about the Strcmp function visit <https://es.mathworks.com/help/matlab/ref/strcmp.html>

¹¹ For more information about the Strcat function visit <https://es.mathworks.com/help/matlab/ref/strcat.html>

¹² For more information about the For loop visit <https://es.mathworks.com/help/matlab/ref/for.html>

¹³ For more information about the If/elseif/else loop visit <https://es.mathworks.com/help/matlab/ref/if.html>

¹⁴ For more information about the Try/catch loop visit <https://es.mathworks.com/help/matlab/ref/try.html>

¹⁵ For more information about the Switch function visit <https://es.mathworks.com/help/matlab/ref/switch.html>

APP DESIGNER

MATLAB App Designer is a program that allows the development of professional applications without having to be an expert programmer in the field of software design.

The application development process can be divided into two sections, the one that corresponds to the development of the code and the part that focuses on the visual part of the application. Both parts are of vital importance since, on the one hand, the developed code must be optimal and error-free to ensure the correct functioning of the application and, on the other hand, the visual part must attract the user's attention and be intuitive to facilitate its use.

For the graphical development of the application, App Designer provides a blank canvas in which the creator can add all desired elements, such as buttons, tables, drop-down lists, among many others that will be described later, by dragging them to the canvas. In addition, the size and appearance of both the canvas and the different elements can be modified by means of an integrated editor that automatically creates the code corresponding to the desired settings. Another advantage of this program is that almost all the components provided have associated functions called *Callbacks*, which are executed when the user interacts with the app. On the other hand, the detection of basic errors during the code development is performed instantly using Code Analyzer, which allows to speed up the process.

App Designer provides both simple and very specific components, including content and instrumentation components. The following is a brief summary of the main elements used.

Button

This component responds when the user presses and releases it. Its appearance can be modified by changing different properties, e.g., its size, color, font, etc.¹⁶



Figure 4. Button component

In the development of this project, this component has been used to select the directory, to choose the display and processing option, i.e., to choose between displaying the tuning curves and the trajectories, as well as to select the days, phases and target cells to be used.

¹⁶ For more information about the Button component visit <https://es.mathworks.com/help/matlab/ref/matlab.ui.control.button-properties.html>

Label

This component contains static text, that is, text that cannot be modified during the execution of the app, and is used to identify parts of an app, which serves as a guide for the user.¹⁷

During this project, labels have been used to identify lists, determining those elements that the user must choose and thus, facilitating the use of the interface. In addition, this component has also been used to identify the lists where the selected items are included.

List Box

The main function of this component is to display items in a list. In addition, it allows the creator to determine whether to activate the Multiselect option or not, which would allow the user to select more than one item in the list. A very useful property when developing an application, is that, as in all the elements provided by App Designer, the size of this component can be varied, in order to adjust it properly to the dimensions of the app and the desired layout. In addition, a scrollbar is automatically created to enable the visualization of all the elements if the list has more components than those that are visible due to the determined size.

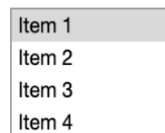


Figure 5. List Box component

This component has been used to display the different options of days, phases and target cells and also the selected items. It should be noted that duplicate elements are allowed, which needs to be taken into account in order to avoid repetition of elements, and that only string type items are allowed, so different functions such as *string* have been needed in order to adjust the data to the type of elements allowed in this component.¹⁸

¹⁷ For more information about the Label component visit <https://es.mathworks.com/help/matlab/ref/matlab.ui.control.label-properties.html>

¹⁸ For more information about the List Box component visit <https://es.mathworks.com/help/matlab/ref/matlab.ui.control.listbox-properties.html>

Text Area

This component's main function is to display multiple lines of text and has been used in the development of this GUI to determine the number of target cells available. In addition, the Editable option has been deselected to prevent the user from modifying this number. The advantage of this component over the use of a label is that it has a label associated with the text area, which makes it easier to identify what is being displayed by this component.

Panel

As for the container components used in this project, only the Panel component has been used. This allows grouping different components, which provides a clearer and more orderly visual effect of the application, facilitating its use.¹⁹

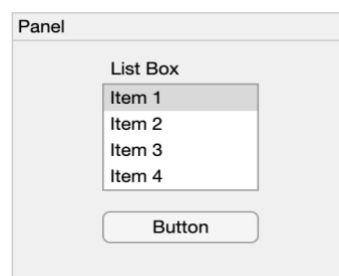


Figure 6. Panel component

The use of panels in the development of this app has allowed the separation and visual distinguishment between the three parts corresponding to the selection of days, phases, and target cells to study.

Finally, it should be noted that other elements have been used such as the Table component, which has allowed to visualize the outputs with matrix or vector structure of the functions used and mentioned in section 5.2.2., allowing to compare the results obtained by both in MATLAB and App Designer. This has been very useful when developing this project since it has been necessary to modify the code due to the fact that the functioning and code structure of App Designer is not exactly the same as in the MATLAB Editor. The Edit Field component has also been used to view the iterations as the code is executed and determine during which iteration an error occurred.

¹⁹ For more information about the Panel component visit <https://es.mathworks.com/help/matlab/ref/matlab.ui.container.panelappd-properties.html>

5.2.4. Design options

As aforementioned, the main objective of this project is to develop a graphical user interface to process and visualize the data obtained from experiments related to Brain Computer Interfaces. Specifically, the developed app allows to obtain and visualize the tuning curves of the selected target cells, enabling the user to see the changes that occur in the spiking rates of the neurons and determine the angle of preference of each of them according to the day. In addition, this graphical interface allows the visualization of the trajectories performed in each of the experiments performed by the subject under study, allowing to determine if there were variations in the linearity of these, which would indicate an advance in learning.

Throughout this project, two different versions have been developed. In the original version, it was considered that the application should have a main screen in which the user could select the directory where the data of interest is located. In addition, in this same screen, the user also had to choose the experiment phase to visualize. In this case, it was only allowed to select a single phase, as it was considered that this would simplify the application. Finally, the target cell had to be chosen, as shown in the image below.

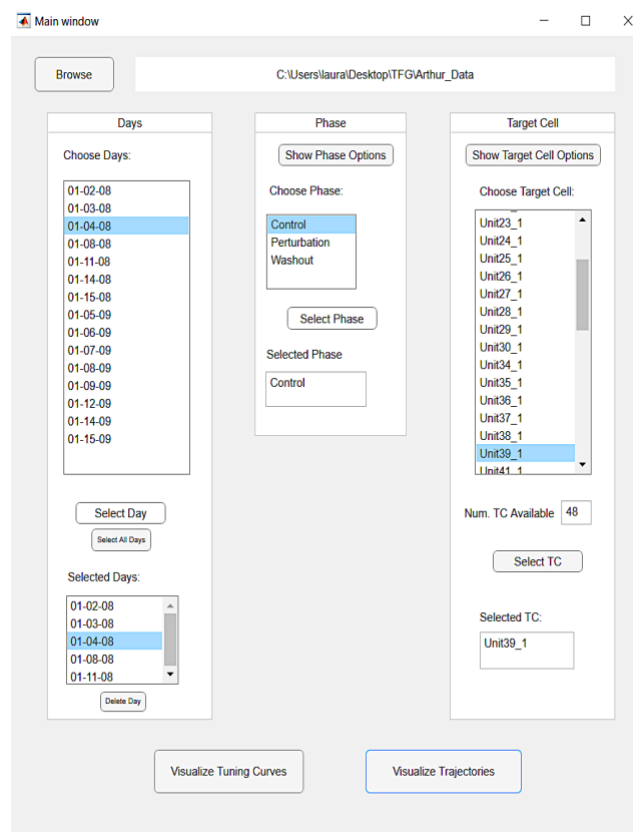


Figure 7. Main window from the first version

At the end of this window, two buttons executed the code of two independent windows, one specific to visualize the tuning curves and the other to show the trajectories performed by the subject for the given days.

Figure 8 shows the window generated by interacting with the Visualize Tuning Curves button. The data selected by the user could be seen in the upper part of the window, and then the tuning curves of the selected days were displayed vertically and in descending order.

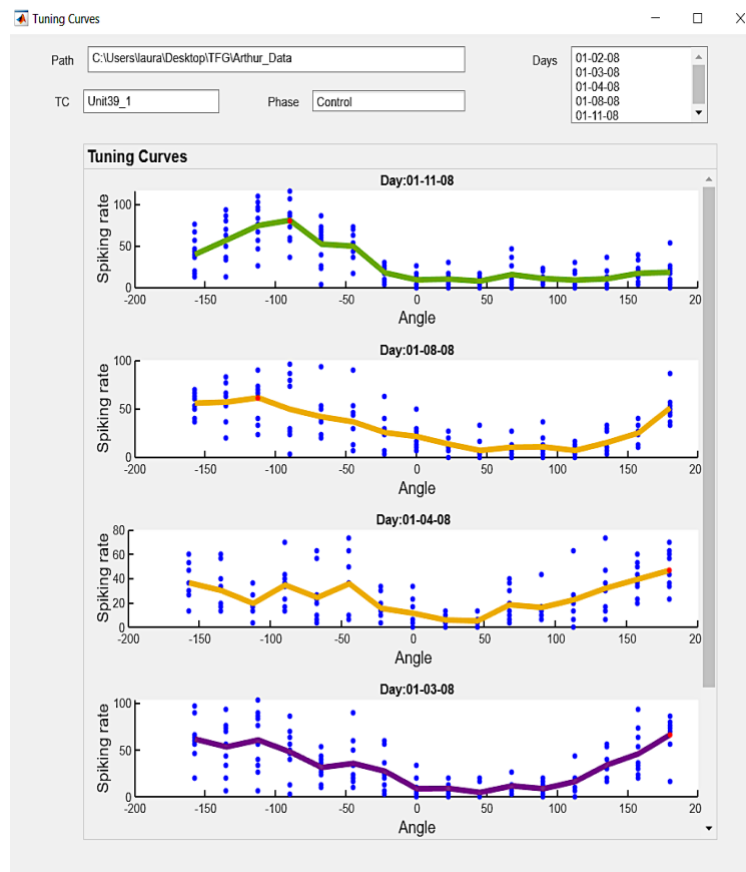


Figure 8. Tuning Curves window from the first version

As for the screen corresponding to the trajectory display, which can be seen in Figure 9, it was decided to show only the trajectories performed by the subject during the first experiment of each day.

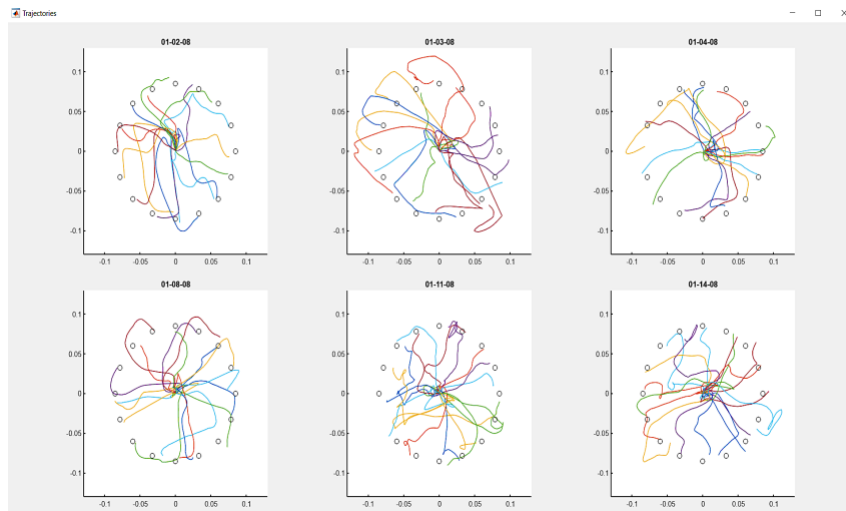


Figure 9. Trajectories window from the first version

After finishing this first version of the application, an analysis of the final aspect and the level of usefulness of the information provided by this interface was performed, and it was concluded that certain aspects should be modified in order to provide the user with a greater amount of information, which would allow the comparison of more relevant data and thus make it possible to draw a greater number of conclusions.

For this purpose, it was decided that when viewing the tuning curves, it would be more convenient to compare results from different phases, rather than only displaying results from a single stage, as this would facilitate the determination of whether the spiking rates of a target cell vary depending on the phase of the experiments under study. In the case of trajectories, however, it was thought that being able to visualize all trajectories from each day's experiments would be preferable for defining linearity changes between trials. In addition, it was decided that, depending on the number of days selected, the size of the panels where the different trajectories were displayed would be modified, so that the maximum number of experiments would be visible.

Since for each option the user must choose different factors, it was decided that, in the main screen, only the option to select the directory of the folder where the data to be studied are collected will be shown, and that the rest of the parameters will need to be set in the following windows. In the case of the option Visualize Tuning Curves, the user will be able to choose more than one phase, in order to allow the comparison of the results, and it was decided that additional windows would be added for each selected phase to show the preference angles and the corresponding spiking rates obtained each day.

5.2.5. Graphical User Interface Development

This section will describe in detail the process of creating each of the windows that compose the interface.

As it has been previously explained, the functioning of this application is based on the selection of the desired data and the necessary parameters depending on the objective to be achieved.

The developed interface has a main window, as mentioned in the previous section, in which the user must select the folder where the data of the experiments are stored. It is essential to highlight the need for this data to have exactly the structure mentioned in section 5.2.1., otherwise the program will not be able to execute the code correctly.

Once the directory has been selected, the user must choose the option of interest, either to visualize the tuning curves or the trajectories. In both cases, a window will appear showing only the available days, and two buttons to select a specific day or all days at once, which can speed up the use of the application. Once the days have been chosen, a list will appear showing the selected elements, where it is possible to delete a specific date, or all of them in case the user wants to start again with the selection of days. As for the phase options (control, disturbance and washout), they will be displayed with a single button to select the desired option, in the case of the trajectories window since only one phase can be selected, or with two buttons, one to select phases one at a time, and another to select all phases at once, in the case of the tuning curves window since it has been determined that allowing multi-selection of phases would increase usability. Additionally, a list of selected phases will be provided, where it will be possible to delete the non-desired phases that the user finally decides to discard, in case the option to visualize the tuning curves has been selected.

As in the case of the days, once the phases are selected, a list of common target cells for the selected days and phases will be displayed to ensure that the execution and comparison of the data are possible. After specifying the target cell of interest, the tuning curves and trajectories corresponding to the given parameters will be automatically displayed, depending on the option chosen.

Next, the code developed for the creation of each window will be explained, differentiating between properties, startup functions and callback functions. In addition, at the beginning of each section, the design of the window and its components will be described.

MAIN WINDOW

When executing the application, the user is presented with the main window, which is the simplest of all the graphical interface since it is composed of only three buttons and a label. In the center, a button to select the desired folder, located next to a label, which will show the specified directory, can be found. At the bottom, the two buttons that will redirect the user to the desired window can be seen. Each button has been renamed to facilitate its location, as shown in Figure 10.

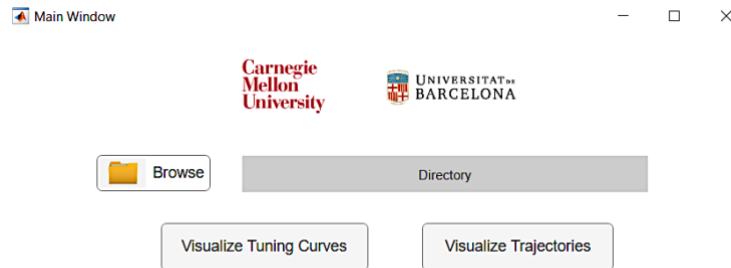


Figure 10. Main window

In addition, the logos of the two universities that have made possible the realization of this project, Carnegie Mellon University and the University of Barcelona, can be seen at the top of the window. For this purpose, two Image components have been used, in which it has been necessary to modify some parameters such as position and size to adjust the images correctly.

As for the names that have been defined for each component, the image components where the logos are displayed have been named as `app.ImageCMU` and `app.ImageUB`, and the buttons that redirect the user to the following windows have been called `app.VisualizeTuningCurvesButton` and `app.VisualizeTrajectoriesButton`. In addition, the name of the button that allows selecting the folder and the name of the label where the selected directory is shown have been changed to `app.BrowseButton` and `app.DirectoryLabel`, respectively.

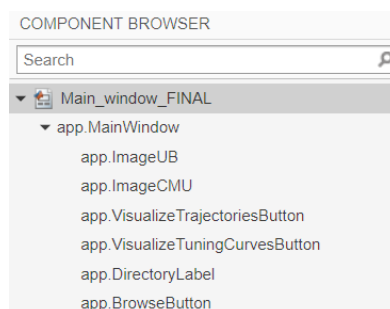


Figure 11. Main window components

It should be noted that when inserting the components in the canvas, App Designer automatically generates the necessary code for the creation of the components, where the established design parameters are reflected.

Properties

When generating an application in App Designer, it is defined with a class, to which correspond a series of properties and methods. For the development of this window it has been necessary to use three properties, two that refer to the following windows that will be generated when clicking on the Visualize Tuning Curves and Visualize Trajectories buttons, and another property that will be used to share the directory selected in this window with the following ones.

```
properties (Access = private)
    selectedPath %Selected directory
    callerApp % App Tuning Curves Window
    callerApp_2 % App Trajectories Window
end
```

Figure 12. Main window properties

Startup Function

The execution of this function occurs every time the application is opened. The variable `app` refers to the application that is running, in this case `Main_window_FINAL.mlapp`. As can be seen in Figure 13, when this function is executed, the name of the window is set, along with the source of the two logo images.

```
26 function startupFcn(app)
27
28     app.MainWindow.Name = 'Main Window';
29     app.ImageCMU.ImageSource = 'CMU.png';
30     app.ImageUB.ImageSource = 'UB.jpg';
31
32 end
```

Figure 13. Main window Startup function

Callback functions

This type of function refers to those that are executed when the user interacts with the application components. As this window presents three buttons, the generation of a callback function for each one of them is necessary. They can be found in their entirety in Annex I.

In Figure 14, the function that determines the behavior of the Browse button can be seen.

```

33 function BrowseButtonPushed(app, event)
34
35     app.selectedPath=uigetdir;
36     app.MainWindow.Visible = 'off';
37     app.MainWindow.Visible = 'on';
38     app.DirectoryLabel.Text=app.selectedPath;
39
40 end
    
```

Figure 14. Browse button callback function

As explained above, this button is used to allow the user to select the desired folder. For it, the *uigetdir* function has been used, which allows obtaining the directory of the desired location, which is saved in the `app.selectedPath` property so that it can be used in the next windows. The following two lines of code have been used to avoid the window to be sent to the background after selecting the folder of interest, and the last line sets the label text from the `app.DirectoryLabel` to be the selected path.

Finally, the following two functions observed in Figure 15, correspond to the callback functions of the buttons that redirect to the secondary windows.

```

45 function VisualizeTuningCurvesButtonPushed(app, event)
46
47     app.callerApp = TC_FINAL(app.selectedPath);
48
49 end
50
51 % Button pushed function: VisualizeTrajectoriesButton
52 function VisualizeTrajectoriesButtonPushed(app, event)
53
54     app.callerApp_2 = Trajectories_FINAL(app.selectedPath);
55
56 end
57 end
    
```

Figure 15. Visualize Tuning Curves button and Visualize Trajectories Button callback functions

In both lines of code, 47 and 54, the corresponding application of the window to be opened is called, adding as input argument the path selected by the user and setting as output the objects `app.callerApp` and `app.callerApp2` defined in Properties.

TUNING CURVES WINDOW

In this window, the user can visualize the tuning curves of interest. For this purpose, three independent panels have been created to facilitate selecting the days, the phases, and the target cell to be studied. As explained above, this selection will be made gradually to guide the user and prevent errors. This means that, when initializing the window, not all three panels visible in the upper part of Figure 16 will be observed. First, only the day panel will be accessible, and once the desired days have been selected, the phase panel will appear, and the same with the target cell panel. Finally, when the desired neural unit is selected, the tuning curves will be displayed. To this end, a panel will be created for each selected phase, where the obtained curves will be visible.

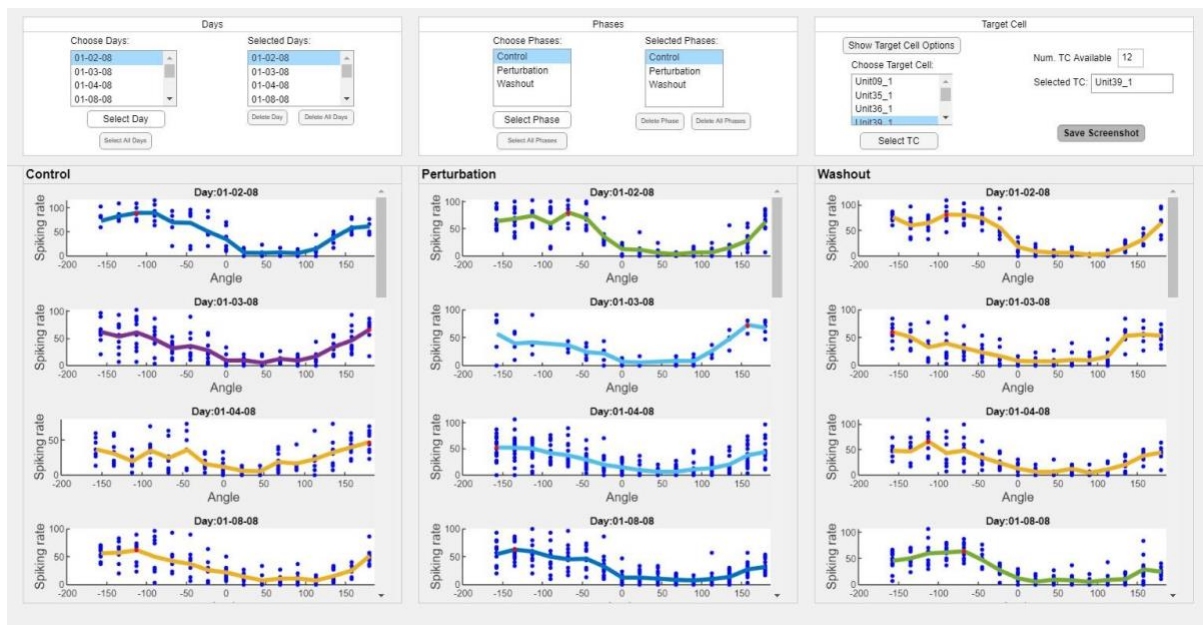


Figure 16. Tuning Curves window

As shown in Figure 17, the first two panels are composed of two list boxes each, the first one where the available options are displayed, `app.ChooseDaysListBox` and `app.ChoosePhaseListBox`, and the second one where the selected items are exposed, `app.SelectedDaysListBox` and `app.SelectedPhasesListBox`. In addition, both contain four buttons, two for each list box. The ones corresponding to the list of options to choose from are the ones that allow the user to select individual items or all items at once. In contrast, the ones below the list of selected items allow the user to delete items one by one or clear the entire list.

As for the third panel, it is composed of three buttons, `app.ShowTargetCellOptionsButton`, which allows displaying the options from which

the user can choose, `app.SelectTCButton`, which saves the selected item in the list of available target cells and executes the option to display the tuning curves, and finally, `app.SaveScreenshotButton`, that allows to generate and save a screenshot in order to allow the user to access the results after closing the application. In addition, two Text Area components have been added in which the number of available units and the chosen option are shown.

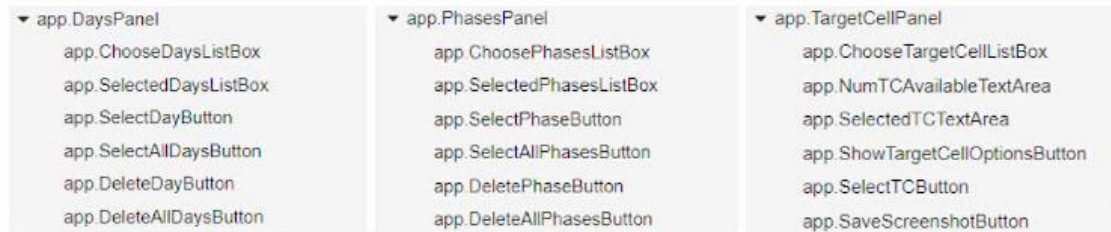


Figure 17. Tuning Curves window components

Finally, it was considered useful to provide the user with an independent window per phase to provide the data of the angle of preference and the respective spiking rate obtained for each day, for a faster comparison of the results. In addition, as can be observed in Figure 18, each of these windows includes a button that allows the user to save these results in a CSV file for further study.

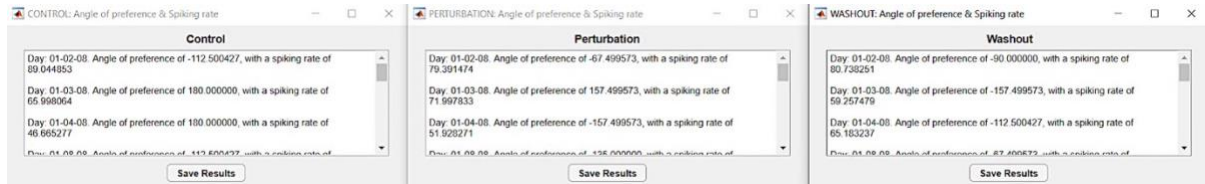


Figure 18. Independent windows showing the angle of preference and its spiking rate

Properties

For the development of this window, a total of 47 properties have been necessary due to the large amount of information that must be shared between the different components of the interface. These variables can be divided into four groups depending on their function during the execution of the window: variables used when selecting the options, when obtaining the target cells list, when generating and plotting the tuning curves and, the variables used to open the windows where the main results are shown. For more information on the properties used, consult Annex II.

Startup Function

The startup function corresponding to this window has an input parameter, `path`, which has been necessary to add in order to access the directory chosen by the user in the main window, which has been stored in the variable `app.selectedPath`. Subsequently,

the names of the folders found in this directory have been extracted, which correspond to the days on which the experiments were carried out. On the other hand, it has been verified that the file *file_inds.mat* is accessible in all the folders and, in order to be able to show the dates in ChooseDaysListBox in chronological order, it has been necessary to convert the names of the folders to scalar datetime arrays using the *datetime* function. It was necessary to specify "MM-dd-yy" as the output format to match the one used when the folders were created. Finally, control, perturbation, and washout were set as possible phases to be chosen by the user, in addition to the title of the window, which was named "Tuning Curves" using the *app.TCWindow.Name* function. It was also established that when starting the window only the available days and the corresponding selection buttons would be visible.

Callback functions

Due to the large number of components that make up this window, only the most relevant functions that have been used will be mentioned in this section. To consult the complete code used for the development of this window, refer to Annex II.

ShowTargetCellOptionsButtonPushed(app, event)

This function allows showing the common target cells for all the days and phases selected by the user, when interacting with this button. This is very important in order to compare the evolution of the behavior of a target cell for the different days and phases since, if the user chooses a unit whose data have not been collected during the development of the experiments of interest, the desired comparison would not be possible.

For this purpose, the identification number of the cells studied in each experiment has been extracted using the *fieldnames* function, and subsequently, the *intersect* function has been used in order to obtain a vector formed only by the cells common to the selected day's experiments.

```

730 -
731 -
732 -
733 -
734 -
735 -
736 -
737 -
738 -
739 -
740 -
741 -
742 -
743 -
744 -
745 -
746 -
747 -
748 -
749 -
750 -
751 -
752 -
753 -
754 -
755 -
756 -
757 -
758 -
759 -
760 -

if strcmp('control', phase_sel)
    path_control = d.(Const{i});
    path_length_control = length(path_control);
    for file = 2:path_length_control
        try
            if isempty(path_control)
            else
                app.id_control = num2str(path_control(file));
                if numel(app.id_control) == 4
                    start_fname = 'Arthur_BC.0';
                else
                    start_fname = 'Arthur_BC.';
                end
                path_file_control = strcat(string(app.selectedPath), '\', string(days(day_1)), '\', start_fname, num2str(path_control(file)), end_fname);
                e_control = load(path_file_control);
                const1_control = fieldnames(e_control); % Variables that are included in this file
                struct_spikes_control = e_control.(const1_control{6}); % Spikes variable
                t_cells_control = fieldnames(struct_spikes_control); % List of all the units
                app.C_control{end+1} = t_cells_control(1:end-1);
                app.uvals_control = app.C_control{i};
                % Keep only the common target cells
                for i_control = 1:numel(app.C_control)
                    app.uvals_control = intersect(app.C_control{i_control}, app.uvals_control); % Vector with the common target_cells
                end
            end
        catch err
            %errordlg(err.message)
        end
    end
end

```

Figure 19. Code used to obtain the common cells list

Figure 19 shows the code used to obtain the total number of common target cells for the control phase. On the other hand, a distinction was made according to the situations that could occur since, if only one phase is selected by the user, the resulting vector of target cells does not have to go through any other procedure, unlike if several phases are selected. In this latter case, after obtaining the vector of each of the specified phases, the *intersect* function needs to be applied to obtain only the target cells belonging to all the vectors.

Finally, only the target cells with name ending in 1 are selected, since they correspond to the valid units.

```

958 -
959 -
960 -
961 -
962 -
963 -
964 -
965 -
966 -
967 -
968 -
969 -
970 -
971 -
972 -
973 -
974 -
975 -
976 -
977 -
978 -
979 -
980 -
981 -
982 -
983 -
984 -
985 -
986 -
987 -
988 -
989 -
990 -
991 -
992 -
993 -
994 -
995 -
996 -
997 -
998 -
999 -

%If more than one phase is selected, the common target cells
%for all the phases need to be extracted
elseif app.a == 2
    if ismember('Control', phase_sel(1)) || ismember('Control', phase_sel(2))
        if ismember('Perturbation', phase_sel(1)) || ismember('Perturbation', phase_sel(2))
            u = app.uvals_control;
            u1 = {};
            for i = 1:numel(app.uvals_perturbation)
                u2 = char(intersect(app.uvals_perturbation(i), u));
                if isempty(u2)
                else
                    u1{end+1} = u2;
                end
            end
            app.target_cells_list = [];
            for i = 1:length(u1)
                item = char(u1(i));
                if str2double(item{end}) == 1
                    app.target_cells_list = [app.target_cells_list convertCharToStrings(item)];
                end
            end
        else
            u = app.uvals_control;
            u1 = {};
            for i = 1:numel(app.uvals_washout)
                u2 = char(intersect(app.uvals_washout(i), u));
                if isempty(u2)
                else
                    u1{end+1} = u2;
                end
            end
            app.target_cells_list = [];
            for i = 1:length(u1)
                item = char(u1(i));
                if str2double(item{end}) == 1
                    app.target_cells_list = [app.target_cells_list convertCharToStrings(item)];
                end
            end
        end
    end
end

```

Figure 20. Generated code to obtain the final target cell list

SelectTCButtonPushed(app, event)

The main purpose of this window is to display the tuning curves corresponding to the options selected by the user in order to compare the subject's neural activity on different days and phases. This has been possible mainly thanks to this function, which is executed once the user has selected the cell of interest. For the visualization of the graphs it has been considered that the use of one panel per phase is the most logical layout for visual comparison. With this in mind, the size of the panels has been set so that, depending on the number of phases selected, they are more or less wide, to optimize the use of the available space. In addition, it has been determined for the panels to have a scrollable bar so that the user can visualize the totality of the curves in case of selecting several days.

The base function used to obtain the desired graphs was *Visualize_tuning_curves*, described in section 5.2.2., by means of which the rates and angles extracted from the data of each experiment are obtained and plotted. The subfunctions used to obtain these values have been described as methods in the application, which allows them to be called from different places in the code.

SaveScreenshotButtonValueChanged(app, event)

As mentioned earlier, a button has been provided so that the user can generate and save a screenshot of the window, in order to be able to share and study the visible plots. This is possible thanks to the execution of this function, the code of which can be seen in Figure 21. First of all, it has been established the file formats in which the user can save the image as convenient, and the *uiputfile* function has been used to provide a dialog box in which the user can determine the name of the file and its saving directory. If the file name has the correct format, the *exportapp* function generates and saves the screenshot. In addition, the last two lines of code have been added to prevent the Visualize Tuning Curves window from being sent to the back.

```

2104 function SaveScreenshotButtonValueChanged(app, event)
2105
2106     % Save a screenshot of the window
2107     filter = {'*.jpg'; '*.png'; '*.tif'; '*.pdf'};           % File type options
2108     [filename, filepath] = uiputfile(filter);               % Open dialog box for saving files
2109     if ischar(filename)                                    % If the name has the correct format, save the file
2110         exportapp(app.TCWindow, [filepath filename]);
2111     end
2112     app.TCWindow.Visible = 'off'; % These two lines of code work-around an issue whether the figure is sent to the background.
2113     app.TCWindow.Visible = 'on';
2114
2115 end

```

Figure 21. Code used to generate and save the screenshot

TRAJECTORIES WINDOW

Another valuable aspect to be studied from the data obtained from the experiments carried out by Carnegie Mellon University, is the evolution of the trajectories performed by the subject, since it has been seen that an increase in the linearity of the trajectories is indicative of an advance in the learning process. For this reason, a window has been developed to allow the user to visualize the trajectories of all the experiments performed in a day, as well as giving the option to visualize several days at a time. When developing this window, it was considered to be more useful allowing the visualization of several days' data for a single phase than displaying trajectories for several phases but for a single day. Therefore, the three-panel structure used for selecting options in the Visualize Tuning Curves window was retained, but in this case, the user is only allowed to select a single phase. For this reason, the buttons for selecting all phases and deleting those already selected have been suppressed. Despite this, the basic functionality of both windows is the same, i.e., the panels will appear progressively as the user selects the options.

To visualize the different trajectories, it has been decided to create one panel per selected day that gathers the results of all the experiments performed, as shown in Figure 22. Its size will be determined by the number of days to be analyzed, but it has been designed to show as many experiments as possible.

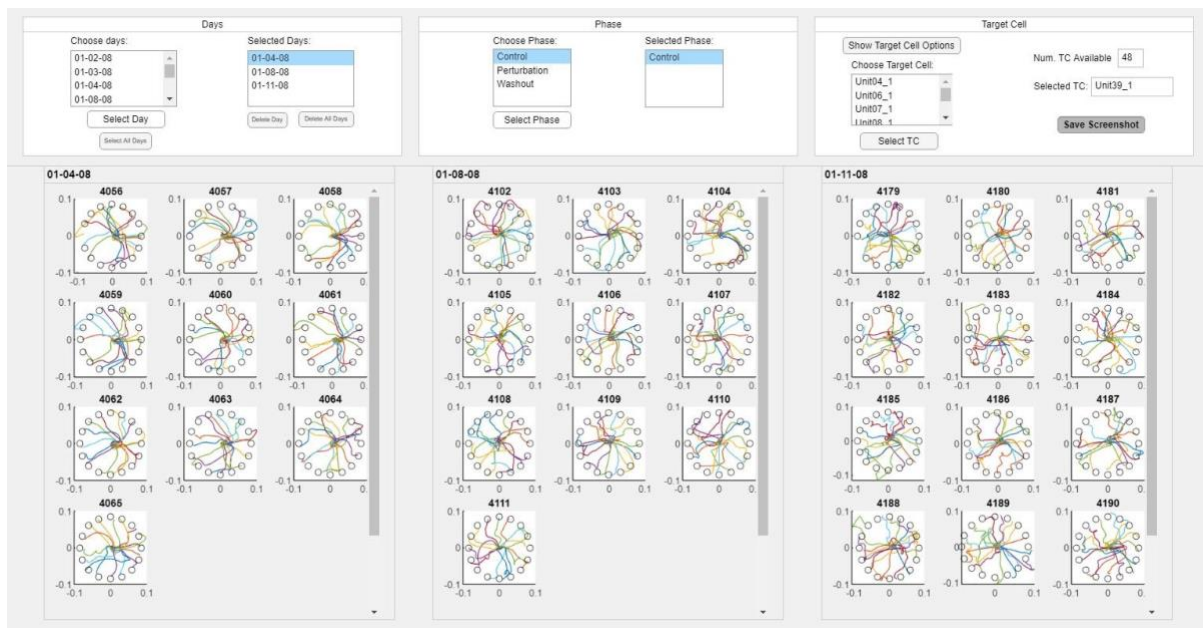


Figure 22. Trajectories window

Properties

For the development of this window, significantly fewer properties have been required compared to those needed for the Visualize Tuning Curves window, since the acquisition of trajectories and targets is much simpler. A total of 13 properties have been created to share information between the different components of the window. Among them, it is worth mentioning the two variables that collect the information about target positions and trajectories, which have been called `app.targs` and `app.trajs`. The rest of the variables created are used to store the options selected by the user and to define the list of available target cells. For a detailed description of the properties of this window, refer to Annex III.

Startup Function

The startup function of this window coincides with the one used in the application to visualize the tuning curves, since in this window it is also necessary to obtain the available days that correspond to the names of the folders where the files of each experiment are stored. It is also required to ensure that the `file_inds.mat` file, located in each folder, is accessible in order to know the identification number of the files corresponding to each phase of the experiments.

Callback functions

In this section, the operation mode of the two most important callback functions generated during the development of this app will be explained. To consult the rest of the callback functions corresponding to the remaining window components, refer to Annex III.

ShowTargetCellOptionsButtonPushed(app, event)

The objective and structure of this function are very similar to the one used for the Tuning curves window as the intention is the same, to obtain a vector with the target cells common to the established options. In this case, the code has been significantly reduced because, since the user is not allowed to select more than one phase, it is only necessary to consider the case of a specific stage of the experiment. Therefore, only the common target cells for the files of the selected days have been determined in the case of the specified phase, and it has not been necessary to intersect with the vectors obtained for the other options.

SelectTCButtonPushed(app, event)

This function is the one that corresponds to the acquisition and visualization of the trajectories, which is executed when the user selects the target cell of interest.

First of all, as mentioned above, it has been necessary to create a panel per selected day to visualize the results obtained, whose size and location will depend on the number of days chosen. Concerning this, in order to allow the user to visualize as many experiments as possible, it has been determined that if the number of days selected is less than eight, a single row of panels will be available, whose number of columns will be reduced as the number of days increases. On the other hand, if this selection is greater than 8, it has been considered necessary to distribute the panels in two rows, reducing the number of experiments that the user can visualize per day but allowing comparing the experiments of a greater number of days.

Once the panels have been established, the trajectories positions are extracted using the *extract_trajectories* subfunction mentioned in section 5.2.2., for the subsequent plotting of the results obtained using the code provided by the *visualize_trajectories* function. In this case, it has been necessary to include the code of the subfunction inside the main function, due to problems produced by App Designer when trying to call the subfunction when it was defined as a method. This has allowed the use of the *try/catch* function to determine the errors produced in a straightforward way, which has facilitated the adaptation of the code.

APP PACKAGING

Once the development of the graphical interface was completed, the final version was generated using the MATLAB Compiler tool, which allows the application developed in App Designer to be packaged and shared as a standalone desktop application. This tool generates an *.exe* file that limits the use of this interface to computers with Windows operating systems.

6. TECHNICAL VIABILITY

6.1. Technical specifications and characteristics of the software

As mentioned above, the interface generated during this project is packaged as a standalone desktop application limited to the Windows operating system. In case the user does not have this operating system, it will be necessary to install a virtualization software such as Virtualbox to use Windows and access the application.

In addition, regardless of the computer available, it will be necessary to install the free MATLAB runtime tool, specifically version 9.9, since MATLAB version R2020b has been used for this project. This consists of a series of independent libraries that allow the execution of applications generated with MATLAB. Once this program is installed, the user will be able to access the app.

On the other hand, as the software generated is implemented as a set of *.mlapp* files (one for each generated window), another option to share the application would be to compress these files in a zip folder. To read such files, the end user will need to have the MATLAB license fully installed. It should be noted that in this case, the application could be modified by anyone in possession of these files, since the developed code would be fully accessible.

6.2. Strengths, Weaknesses, Opportunities and Threats

In this section, an analysis of the internal (strengths and weaknesses) and external (threats and analysis) characteristics will be carried out.

As can be seen in Table 3, some of the main strengths include a basic knowledge of programming, which has allowed speeding up the development of this project, and the availability of the basic functions to process and visualize the data. As for the main weaknesses, the time limitation to finish the project, and the complexity of the data and functions to be used, in addition to the lack of specific knowledge in the field of BCI and in app development, stand out. It should be noted that these weaknesses have been overcome by devoting a considerable part of the time to understand the data and functions of the experiments on which the project is based, and to become familiar with the tool used to develop the GUI.

Regarding the external analysis carried out, the rapid advancement of BCI technology and the vast experience of the competition in this field stand out among the threats. But, a great number of opportunities have been found, such as the increase in cases of mental disabilities, as well as the applications of BCI technology. Finally, it is important to note that in recent years an increase in the concern for health by society has been observed, and it is predicted that this will increase in the future. This opens up a large number of opportunities as this will lead to an increase in the use of this technology.

Strengths <ul style="list-style-type: none">• Basic knowledge of computer programming• Availability of the basic functions	Weaknesses <ul style="list-style-type: none">• Time limitation to finish the project• The complexity of the data and functions to be used• Lack of initial knowledge of the field of brain-computer interfaces and the software
Opportunities <ul style="list-style-type: none">• Increase of cases of mental disabilities• Increase in the number of applications for BCI technology• Increase in the concern for mental health by society	Threats <ul style="list-style-type: none">• Rapid advancement of BCI technology• Vast experience of the competition in this field

Table 3. SWOT analysis

7. TIMING

7.1. Work Breakdown Structure

The WBS scheme for this project is shown below. As it can be seen, in this project there are four main blocks that include the rest of the subtasks. In the first place, there is the study of the BCI technology since, for the correct development of this project, it has been necessary to know everything related to this technology, since the objective of this work is the development of a tool for the visualization and processing of data obtained from experiments carried out with BCI technology. Likewise, a market study has been carried out to know the technological trends in this field and the development of this technology, as well as the existing software currently in use. As for the development of the user interface, first of all the structure and function of the tool to be developed have been established, and a study of the most optimal design options has been carried out. Subsequently, the corresponding software has been developed. Once the programming and design part has been completed, functional and evaluation tests have been carried out to confirm the absence of errors in the program and to determine the degree of usefulness provided by the developed tool when visualizing and processing data related to BCI technology. Later, a revision of the document to be delivered was carried out, after describing the project development in its entirety. As the last phase of the project, the supporting material will be prepared either in Power Point format or video presentation and an oral presentation and defense will be conducted for the evaluation of the project by a jury.

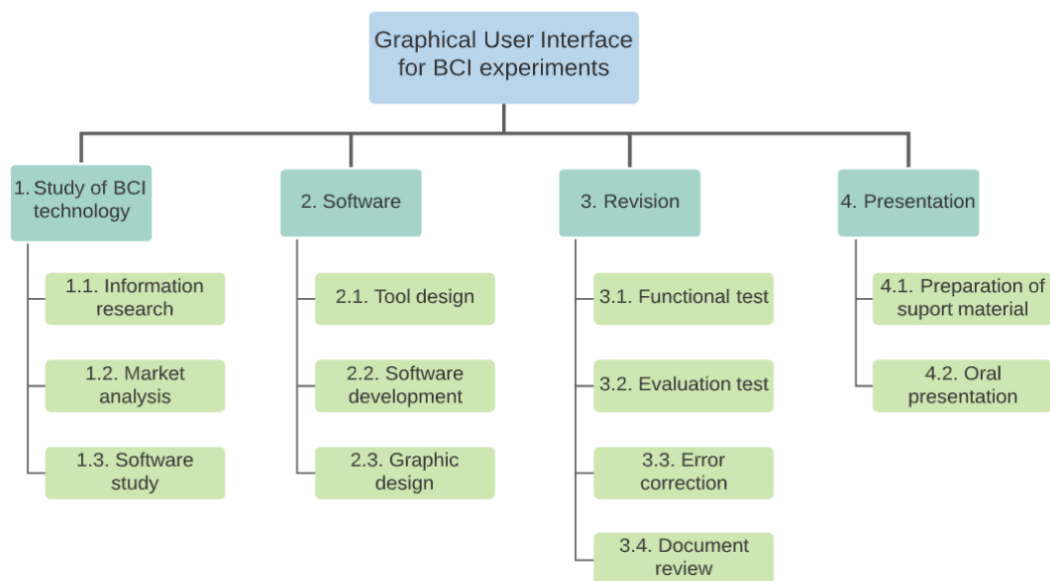


Figure 23. Work Breakdown Structure

7.2. PERT Analysis

Below, a table where it is shown the progress of the project clearly and simply can be seen. The corresponding PERT diagram was developed from the data contained in this table, where the identifiers correspond to the ones used in the previous section. It can be seen the line of work that will be followed throughout the project, as well as the time it will take to develop each activity.

Activity	Identifier	Previous activity	Duration (in days)
Information research	A	-	12
Market analysis	B	-	8
Software study	C	A, B	8
Tool design	D	C	19
Software development	E	D	165
Graphic design	F	D	38
Functional test	G	E, F	14
Evaluation test	H	E, F	18
Error correction	I	G, H	22
Document review	J	I	14
Preparation of support material	K	J	12
Oral presentation	L	K	1

Table 4. Sequence and time matrix of the activities

The PERT diagram obtained after analyzing the above table is shown below. The critical path is highlighted, which is formed by the set of tasks that, if delayed, affect the project's final deadline.

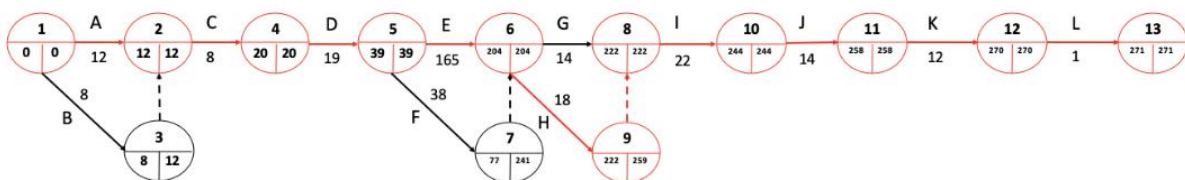


Figure 24. PERT diagram

7.3. GANTT diagram

The GANTT diagram is a graphic tool that allows to establish the course of the project in greater detail. In this diagram it can be observed the deadlines established for each activity, which allows to organize correctly the process of development of the project and thus to be able to meet the established delivery deadlines.

Date \ Activity	15/04/2020	01/05/2020	05/05/2020	17/05/2020	20/05/2020	30/05/2020	10/11/2020	15/11/2020	30/11/2020	20/01/2021	01/03/2021	25/04/2021	30/04/2021	08/05/2021	21/05/2021	25/05/2021	30/05/2021	14/06/2021	20/06/2021		
1.1. Information research	█																				
1.2. Market analysis		█																			
1.3. Software study			█																		
2.1. Tool design						█															
2.2. Software development							█														
2.3. Graphic design									█												
3.1. Functional test											█										
3.2. Evaluation test												█									
3.3. Error correction													█								
3.4. Document review														█							
4.1. Preparation of support material																			█		

Table 5. GANTT diagram

8. CONCLUSIONS AND FUTURE TRENDS

Technically, knowing the tuning curves of the neural units studied and being able to observe if they undergo any variation during the learning process could allow the determination of the cells involved in this process and to understand the level of brain plasticity of the subject, in order to determine the limits of learning. In addition, being able to compare the trajectories performed during the different trials could help to determine the level of learning of the subject, since it has been seen that an increase in the linearity of the trajectories implies an improvement in learning.

For this reason, it has been decided to develop a graphical user interface that allows researchers to visualize the curves of neural activity as a function of the angle of the target, along with the trajectories performed by the subject to be studied. To do this, an exhaustive study of the experiments on which this project is based, which were conducted by Carnegie Mellon University, was carried out in order to understand the neural functioning of the learning process, and to comprehend the methodology followed for the development of the experiments in question. This last point was of utmost importance since it allowed to determine the parameters or functions that the application to be developed should show.

For the programming and creation of the graphical interface, the MATLAB programming language was chosen since the functions generated during the experiments were developed with this language, in addition to the fact that this language has an app development environment that allows the development of graphical interfaces without the need of being an expert in software design. This resulted in the implementation of a multi-window application composed of three main windows, which allow the user to visualize the tuning curves and trajectories from the data of the experiments determined in the main window. Therefore, despite the difficulties encountered throughout the development of this project, it can be concluded that the main objective has been successfully achieved.

In addition, it has been concluded that this type of application could help in many advances in the medical field, such as rehabilitation after a stroke, as it could be used to determine whether a person will be able to relearn certain movements, which could avoid demotivation and reduce the suffering of the patient in cases where the individual does not have the ability to recover from the lesions. In addition, in the prosthetic sector it would also be useful to determine the neural cells responsible for limb movements and thus be able to determine where to place the electrodes in a

more optimal way, as well as to see how the patient's neural activity evolves during the learning process and determine how long it will take the patient to learn to use the prosthesis in question.

Finally, as possible lines of improvement of the project, the following are proposed:

- Program optimization: although the application is capable of allowing the user to visualize and process experimental data, there are different elements that could be improved to achieve greater usability and efficiency in the use of the interface. As an example of an aspect that has not been possible due to lack of time, the code reduction of the Visualize Tuning Curves window could be considerably reduced to minimize execution time. Testing with other databases and reducing the specificity of the structure of the experiment results could also be done to make it more accessible to other research groups. Another aspect that may slow down the use of the application would be the requirement for the user to use the Show Target Cells button to display and select the unit of interest, when modifying the days and phases of interest. One solution would be by automating the update of the list of available target cell options when modifying the selection of the other parameters. Finally, it could be tested with different users to determine the benefits and constraints of the application, so that improvements could be made as necessary.
- Multiplatform application: as mentioned in this document, the developed user interface is only available for users with Windows operating system, which considerably limits the market sector to which it is addressed. Therefore, the application could be implemented to be independent of the operating system to be used.
- Saving the results : An improvement that could be useful for the user would be to allow, not only to take a screenshot of the results as has been done in this project, but also to save all the graphs generated in an external file to be able to share or study the results afterwards. In addition, the option to save the selected parameters could be generated to be able to replicate the results at another time without having to repeat the whole process, as well as allowing to collect the data used to generate the tuning curves and trajectories and save them in a CSV file.

9. BIBLIOGRAPHY

1. Arafat, I. (2013). Brain - Computer Interface: Past, Present & Future. *International Islamic University Chittagong (IIUC), Chittagong, Bangladesh*, 1–6. Retrieved from https://www.academia.edu/1365518/Brain_Computer_Interface_Past_Present_and_Future
2. Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, 2, 157–180. <https://doi.org/10.1146/annurev.bb.02.060173.001105>
3. Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., ... Flor, H. (1999). A spelling device for the paralysed. *Nature*, 398(6725), 297–298. <https://doi.org/10.1038/18581>
4. Lebedev, M. A., & Nicolelis, M. A. L. (2006). Brain-machine interfaces: past, present and future. *Trends in Neurosciences*, 29(9), 536–546. <https://doi.org/10.1016/j.tins.2006.07.004>
5. Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., ... Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099), 164–171. <https://doi.org/10.1038/nature04970>
6. Chandani R. Suryawanshi, V. N. (2013). Blue Brain. *Journal of Advances in Chemistry*, 10(1), 2146–2161. Retrieved from https://www.researchgate.net/publication/331085055_BLUE_BRAIN
7. Jiang, L., Stocco, A., Losey, D. M., Abernethy, J. A., Prat, C. S., & Rao, R. P. N. (2019). BrainNet: A Multi- Person Brain-to-Brain Interface for Direct Collaboration Between Brains. *Scientific Reports*, 9(1), 1–11. <https://doi.org/10.1038/s41598-019-41895-7>
8. Musk, E. (2019). An Integrated Brain-Machine Interface Platform With Thousands of Channels. *Journal of Medical Internet Research*, 21(10), 12. <https://doi.org/10.2196/16194>
9. Jantz, J., Molnar, A., & Alcaide, R. (2017). A brain-computer interface for extended reality interfaces. *ACM SIGGRAPH 2017 VR Village, SIGGRAPH 2017*, (July), 2. <https://doi.org/10.1145/3089269.3089290>
10. Brunner, C., Andreoni, G., Bianchi, L., Blankertz, B., Breitwieser, C., Kanoh, S., ... Müller-Putz, G. R. (2012). *BCI Software Platforms*. 303–331. https://doi.org/10.1007/978-3-642-29746-5_16
11. Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., & Wolpaw, J. R. (2004). BCI2000: A general- purpose brain-computer interface (BCI) system. *IEEE*

- Transactions on Biomedical Engineering*, 51(6), 1034–1043.
<https://doi.org/10.1109/TBME.2004.827072>
12. Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., ... Lécuyer, A. (2010). OpenViBE: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments. *Presence: Teleoperators and Virtual Environments*, 19(1), 35–53.
<https://doi.org/10.1162/pres.19.1.35>
 13. Abdulkader, S. N., Atia, A., & Mostafa, M. S. M. (2015). Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, 16(2), 213–230.
<https://doi.org/10.1016/j.eij.2015.06.002>
 14. Lin, C. T., Lin, B. S., Lin, F. C., & Chang, C. J. (2014). Brain computer interface-based smart living environmental auto-adjustment control system in UPnP home networking. *IEEE Systems Journal*, 8(2), 363–370.
<https://doi.org/10.1109/JSYST.2012.2192756>
 15. Shin, D., Kim, T., Kim, S., & Shin, D. (2011). Design and implementation of smart driving system using context recognition system. *ISCI 2011 - 2011 IEEE Symposium on Computers and Informatics*, 84–89.
<https://doi.org/10.1109/ISCI.2011.5958889>
 16. Mak, J. N., & Wolpaw, J. R. (2009). Clinical Applications of Brain-Computer Interfaces: Current State and Future Prospects. *Bone*, 23(1), 1–7.
<https://doi.org/10.1038/jid.2014.371>
 17. Lin, C. T., Tsai, S. F., & Ko, L. W. (2013). EEG-based learning system for online motion sickness level estimation in a dynamic vehicle environment. *IEEE Transactions on Neural Networks and Learning Systems*, 24(10), 1689–1700.
<https://doi.org/10.1109/TNNLS.2013.2275003>
 18. Sharanreddy, M., & Kulkarni, P. K. (2013). Automated EEG signal analysis for identification of epilepsy seizures and brain tumour. *Journal of Medical Engineering and Technology*, 37(8), 511–519. <https://doi.org/10.3109/03091902.2013.837530>
 19. Ang, K. K., Guan, C., Chua, K. S. G., Ang, B. T., Kuah, C., Wang, C., ... Zhang, H. (2010). Clinical study of neurorehabilitation in stroke using EEG-based motor imagery brain-computer interface with robotic feedback. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, (August 2010), 5549–5552. <https://doi.org/10.1109/IEMBS.2010.5626782>
 20. Lotte, F., Nam, C. S., Nijholt, A., Lotte, F., Nam, C. S., & Nijholt, A. (2018). Introduction: Evolution of Brain-Computer Interfaces. *CRC Press*, 1–8.
<https://doi.org/10.1201/9781351231954>
 21. Işcan, Z., & Nikulin, V. V. (2018). Steady state visual evoked potential (SSVEP) based brain-computer interface (BCI) performance under different perturbations. *PLoS ONE*, 13(1), 1–17. <https://doi.org/10.1371/journal.pone.0191673>

22. Arafat, I. (2013). Brain – Computer Interface: Past, Present & Future. *International Islamic University Chittagong (IIUC), Chittagong, Bangladesh*, 1–6.
23. K P Naveen Reddy, Geyavalli Y, Sujani D, R. S. M. (2018). Comparison of Programming Languages: Review. *International Journal of Computer Science & Communication*, 9(2), 113–122.
24. Oby, E. R., Golub, M. D., Hennig, J. A., Degenhart, A. D., Tyler-Kabara, E. C., Byron, M. Y., ... & Batista, A. P. (2019). New neural activity patterns emerge with long-term learning. *Proceedings of the National Academy of Sciences*, 116(30), 15210-15215.
25. Jarosiewicz, B., Chase, S. M., Fraser, G. W., Velliste, M., Kass, R. E., & Schwartz, A. B. (2008). Functional network reorganization during learning in a brain-computer interface paradigm. *Proceedings of the National Academy of Sciences*, 105(49), 19486-19491.
26. Zhou, X., Tien, R. N., Ravikumar, S., & Chase, S. M. (2019). Distinct types of neural reorganization during long-term learning. *Journal of neurophysiology*, 121(4), 1329-1341.

ANNEX I. Code used for the development of the main window

Properties

```

1  classdef Main_window_FINAL < matlab.apps.AppBase
2
3      % Properties that correspond to app components
4      properties (Access = public)
5          MainWindow          matlab.ui.Figure
6          BrowseButton        matlab.ui.control.Button
7          DirectoryLabel      matlab.ui.control.Label
8          VisualizeTuningCurvesButton matlab.ui.control.Button
9          VisualizeTrajectoriesButton matlab.ui.control.Button
10         ImageCMU             matlab.ui.control.Image
11         ImageUB              matlab.ui.control.Image
12     end
13
14     properties (Access = private)
15         selectedPath %Selected directory
16         callerApp % App Tuning Curves Window
17         callerApp_2 % App Trajectories Window
18     end
19
20
21

```

Startup function

```

25     % Code that executes after component creation
26     function startupFcn(app)
27
28         app.MainWindow.Name = 'Main Window';
29         app.ImageCMU.ImageSource = 'CMU.png';
30         app.ImageUB.ImageSource = 'UB.jpg';
31
32     end

```

Callback functions

```

35     function BrowseButtonPushed(app, event)
36
37         app.selectedPath=uigetdir;           % Calls 'uigetdir' to obtain the directory location from the user
38         app.MainWindow.Visible = 'off';     % These two lines of code work-around an issue whether the figure is sent to the background.
39         app.MainWindow.Visible = 'on';
40         app.DirectoryLabel.Text=app.selectedPath;  |% Sets the label text to be the selected path
41
42     % Button pushed function: VisualizeTuningCurvesButton
43     function VisualizeTuningCurvesButtonPushed(app, event)
44
45         app.callerApp = TC_FINAL(app.selectedPath);
46
47     end
48
49     % Button pushed function: VisualizeTrajectoriesButton
50     function VisualizeTrajectoriesButtonPushed(app, event)
51
52         app.callerApp_2 = Trajectories_FINAL(app.selectedPath);
53
54     end
55
56     end
57

```

Component initialization

```

58
59 % Component initialization
60 methods (Access = private)
61
62 % Create UIFigure and components
63 function createComponents(app)
64
65 % Create MainWindow and hide until all components are created
66 app.MainWindow = uifigure('Visible', 'off');
67 app.MainWindow.Color = [1 1 1];
68 app.MainWindow.Position = [466 312 679 250];
69 app.MainWindow.Name = 'MATLAB App';
70
71 % Create BrowseButton
72 app.BrowseButton = uibutton(app.MainWindow, 'push');
73 app.BrowseButton.ButtonPushedFcn = createCallbackFcn(app, @BrowseButtonPushed, true);
74 app.BrowseButton.Icon = 'IconoCarpeta.jpg';
75 app.BrowseButton.BackgroundColor = [1 1 1];
76 app.BrowseButton.FontSize = 14;
77 app.BrowseButton.Position = [85 109 103 33];
78 app.BrowseButton.Text = 'Browse';
79
80 % Create DirectoryLabel
81 app.DirectoryLabel = uilabel(app.MainWindow);
82 app.DirectoryLabel.BackgroundColor = [0.8 0.8 0.8];
83 app.DirectoryLabel.HorizontalAlignment = 'center';
84 app.DirectoryLabel.Position = [217 108 368 33];
85 app.DirectoryLabel.Text = 'Directory';
86
87 % Create VisualizeTuningCurvesButton
88 app.VisualizeTuningCurvesButton = uibutton(app.MainWindow, 'push');
89 app.VisualizeTuningCurvesButton.ButtonPushedFcn = createCallbackFcn(app, @VisualizeTuningCurvesButtonPushed, true);
90 app.VisualizeTuningCurvesButton.FontSize = 14;
91 app.VisualizeTuningCurvesButton.Position = [143 37 188 43];
92 app.VisualizeTuningCurvesButton.Text = 'Visualize Tuning Curves';
93
94
95 % Create VisualizeTrajectoriesButton
96 app.VisualizeTrajectoriesButton = uibutton(app.MainWindow, 'push');
97 app.VisualizeTrajectoriesButton.ButtonPushedFcn = createCallbackFcn(app, @VisualizeTrajectoriesButtonPushed, true);
98 app.VisualizeTrajectoriesButton.FontSize = 14;
99 app.VisualizeTrajectoriesButton.Position = [300 37 174 43];
100 app.VisualizeTrajectoriesButton.Text = 'Visualize Trajectories';
101
102 % Create ImageCMU
103 app.ImageCMU = uimage(app.MainWindow);
104 app.ImageCMU.Position = [217 168 75 71];
105
106 % Create ImageUB
107 app.ImageUB = uimage(app.MainWindow);
108 app.ImageUB.Position = [330 155 153 97];
109
110 % Show the figure after all components are created
111 app.MainWindow.Visible = 'on';
112 end
113
114 % App creation and deletion
115 methods (Access = public)
116
117 % Construct app
118 function app = Main_window_FINAL
119
120 % Create UIFigure and components
121 createComponents(app)
122
123 % Register the app with App Designer
124 registerApp(app, app.MainWindow)
125
126 % Execute the startup function
127 runStartupFcn(app, @startupFcn)
128
129 if nargin == 0
130 clear app
131 end
132 end
133
134 % Code that executes before app deletion
135 function delete(app)
136
137 % Delete UIFigure when app is deleted
138 delete(app.MainWindow)
139 end
140
141 end

```

ANNEX II. Code used for the development of the Tuning Curves window

Properties

```

1 classdef TC_FINAL < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5         TCWindow          matlab.ui.Figure
6         TargetCellPanel   matlab.ui.container.Panel
7         SaveScreenshotButton matlab.ui.control.StateButton
8         SelectTCButton    matlab.ui.control.Button
9         ShowTargetCellOptionsButton matlab.ui.control.Button
10        SelectedTCTextAreaLabel matlab.ui.control.Label
11        SelectedTCTextArea matlab.ui.control.TextArea
12        NumTCAvailableTextAreaLabel matlab.ui.control.Label
13        NumTCAvailableTextArea matlab.ui.control.TextArea
14        ChooseTargetCellListBoxLabel matlab.ui.control.Label
15        ChooseTargetCellListBox matlab.ui.control.ListBox
16        PhasesPanel        matlab.ui.container.Panel
17        DeleteAllPhasesButton matlab.ui.control.Button
18        DeletePhaseButton  matlab.ui.control.Button
19        SelectAllPhasesButton matlab.ui.control.Button
20        SelectPhaseButton  matlab.ui.control.Button
21        SelectedPhasesListBoxLabel matlab.ui.control.Label
22        SelectedPhasesListBox matlab.ui.control.ListBox
23        ChoosePhasesListBoxLabel matlab.ui.control.Label
24        ChoosePhasesListBox matlab.ui.control.ListBox
25        DaysPanel          matlab.ui.container.Panel
26        DeleteAllDaysButton matlab.ui.control.Button
27        DeleteDayButton    matlab.ui.control.Button
28        SelectAllDaysButton matlab.ui.control.Button
29        SelectDayButton    matlab.ui.control.Button
30        SelectedDaysListBoxLabel matlab.ui.control.Label
31        SelectedDaysListBox matlab.ui.control.ListBox
32        ChooseDaysListBoxLabel matlab.ui.control.Label
33        ChooseDaysListBox  matlab.ui.control.ListBox
34    end
35
36     properties (Access = public)
37
38         %% Variables used when selecting the options
39         selectedPath = '' % Path selected by user in Main window
40         day          = '' % Day selected in ChooseDaysListBox
41         total_days   = '' % All days selected
42         value_day    = '' % Selected day to delete
43         target_cell  = '' % Selected target cell
44         phases       = '' % Available phases
45         phase_selected = '' % Selected phase
46         phase_selected_total = '' % All phases selected
47         value_phase  = '' % Selected phase to delete
48         a = 0        % Number of phases selected
49
50         %% Variables used when generating the target cells list
51         id_control      % Identification number for a control file
52         id_perturbation % Identification number for a perturbation file
53         id_washout      % Identification number for a washout file
54         C_control = {} % All target cells for the control phase
55         C_perturbation = {} % All target cells for the perturbation phase
56         C_washout = {} % All target cells for the washout phase
57         uvals_control % First column of C_control
58         uvals_perturbation % First column of C_perturbation
59         uvals_washout % First column of C_washout
60         target_cells_list = [] % Target cell list shown in ChooseTargetCellListBox
61
62         %% Variables used when generating the tuning curves
63         path_file_control % File directory in control
64         path_file_perturb % File directory in perturbation
65         path_file_washout % File directory in washout
66         cellnames_control % Target cells available for the control phase
67         cellnames_perturb % Target cells available for the perturbation phase
68         cellnames_washout % Target cells available for the washout phase
69         cell_ind_perturb double % Target cell index in perturbation
70         cell_ind_washout double % Target cell index in washout
71         cellnames_valid_control % Valid cellnames (ending in 1) in control
72         cellnames_valid_perturb % Valid cellnames (ending in 1) in perturbation
73         cellnames_valid_washout % Valid cellnames (ending in 1) in washout
74         angle_control % Angles obtained for control from the VTC function
75         angle_perturb % Angles obtained for perturbation from the VTC function
76         angle_washout % Angles obtained for washout from the VTC function
77         rates_control = [] % Spiking rate in control
78         rates_perturb = [] % Spiking rate in perturbation
79         rates_washout = [] % Spiking rate in washout
80         all_rates_control = [] % All spiking rates in control
81         all_rates_perturb = [] % All spiking rates in perturbation
82         all_rates_washout = [] % All spiking rates in washout
83         cell_ind_control double % Target cell index in control
84
85         %% Variables used to open the windows where the sentences of the results are shown
86         all_sentences_control = [] % All sentences (angle of preference + spiking rate) in control
87         all_sentences_perturbation = [] % All sentences (angle of preference + spiking rate) in perturbation
88         all_sentences_washout = [] % All sentences (angle of preference + spiking rate) in washout
89         callerApp_sent_control % Sentences control window
90         callerApp_sent_perturbation % Sentences perturbation window
91         callerApp_sent_washout % Sentences washout window
92    end

```


Methods

```

94 methods (Access = private)
95 function [info,targetset,cellnames,celldata]=get_seq_rate_info_short_win(app, filenames, rangemode, regressmode, spk_window)
96
97     if ~iscell(filenames)
98         filenames={filenames};
99     end
100
101     if ~exist('rangemode','var')
102         rangemode='half_rt'; %Porque half_rt?
103     end
104
105     if ~exist('regressmode','var')
106         regressmode='rate';
107     end
108
109     cellmode='modulated'; ndim=2;
110     [cellnames,celldata]=check_headers_get_names_sf(app,filenames,cellmode,ndim);
111     [targetset,ntrials]=get_sorted_targetset_sf(app,filenames,ndim);
112
113     info=initialize_info(app,ntrials,ndim,length(cellnames));
114     cumntrials=0; cumnsuctrials=0;
115
116     for fi=1:length(filenames)
117
118         load(filenames{fi});
119
120         [alltrialtimes,si]=sort([trials.ComputerTrialTime;fail_trials.ComputerTrialTime;catch_trials.ComputerTrialTime]);
121         trialtypes=[ones(length(trials.ComputerTrialTime),1);2*ones(length(fail_trials.ComputerTrialTime),1);3*ones( ...
122             length(catch_trials.ComputerTrialTime),1)];
123         trialtypes=trialtypes(si);
124
125         if header.IsBrainControl
126             if exist('em_feedback','var')
127                 timestamps=em_feedback.Time;
128                 positionstamps=em_feedback.CursorPosition;
129             elseif exist('em_movement_command','var')
130                 timestamps=em_movement_command.Time;
131                 positionstamps=em_movement_command.Position;
132             end
133         else
134             timestamps=Plaxon2ComputerTime_sf(app,kinematics.PlexonTime,header,synch);
135             positionstamps=kinematics.Markers;% No entiendo esta parte de la funcion
136         end
137
138
139         for j=1:length(trials.ComputerTrialTime)
140
141             [junk,tind]=ismember(trials.ComputerTrialTime(j),alltrialtimes);
142             targ=trials.TargetPos(j,[1:ndim])./norm(trials.TargetPos(j,[1:ndim]));
143             [junk,targnum]=ismember(targ,targetset,'rows');
144
145             info.seq_successful_trial_no(j+cumnsuctrials)=j+cumnsuctrials;
146             info.seq_trial_no(j+cumnsuctrials)=tind+cumntrials;
147             info.targnum(j+cumnsuctrials)=targnum;
148
149             starttime=trials.ComputerTrialTime(j);
150             endtime=trials.ComputerTrialTime(j)+trials.HoldBFinish(j);
151             mvmttimeinds= (timestamps>starttime & timestamps<endtime);
152             mvmttime=timestamps(mvmttimeinds);
153             mvmt=positionstamps(mvmttimeinds,[1:ndim]);
154
155             if norm(mvmt(1,:))>.005
156                 mvmttimeinds=mvmttimeinds(2:end);
157                 mvmttime=timestamps(mvmttimeinds);
158                 mvmt=positionstamps(mvmttimeinds,[1:ndim]);
159             end
160
161             sw=get_spike_window_sf(app,rangemode,j,trials,header,mvmt,mvmttime,ndim,synch,spk_window);
162             info.spike_window(j+cumnsuctrials,:)=sw;
163
164             for i=1:length(cellnames)
165                 try
166                     sp=spikes.(cellnames{i});
167                     foundflag=1;
168                 catch
169                     foundflag=0;
170                 end
171                 if foundflag
172                     switch regressmode
173                         case {'cnt'}
174                             thesesp=sp(find(sp>sw(1) & sp<=sw(2)));
175                             info.rates(cumnsuctrials+j,i)=length(thesesp);
176                         case {'rate'}
177                             thesesp=sp(find(sp>sw(1) & sp<=sw(2)));
178                             info.rates(cumnsuctrials+j,i)=length(thesesp)/diff(sw);
179                             diff(sw);
180                         case {'frac'}
181                             binsize=diff(sw);
182                             info.rates(cumnsuctrials+j,i)=find_frs_fractionalint_onespiketrain_sc...
183                                 (app,sp'*1000,sw(1)*1000,binsize*1000);
184                     otherwise
185                         error(['Unknown regressmode of ',regressmode]);
186
187                     error(['Unknown regressmode of ',regressmode]);
188                 end
189                 else
190                     info.rates(cumnsuctrials+j,i)=0;
191                 end
192             end
193         end
194         cumntrials=cumntrials+length(alltrialtimes);
195         cumnsuctrials=cumnsuctrials+j;
196     end
197     return
198 end

```

```

198 %SUBFUNCTIONS_GET_SEQ_RATE_INFO_SHORT_WIN
199
200
201 function [cellnamemat,celldata]=check_headers_get_names_sf(app,filenames,cellmode,ndim)
202 for fi=1:length(filenames)
203     load(filenames{fi});
204     if fi==1
205         celldata.mds=header.cell_data.ModDepth;
206         celldata.pds=header.cell_data.PD(:,[1:ndim]);
207         celldata.b0s=header.cell_data.BaseRate;
208         celldata.ismodulated=header.cell_data.ModulatedCellsList;
209         celldata.bs=[celldata.b0s,celldata.pds];
210         celldata.bs(:,2)=celldata.bs(:,2).*celldata.mds;
211         celldata.bs(:,3)=celldata.bs(:,3).*celldata.mds;
212         if ndim==3
213             celldata.bs(:,4)=celldata.bs(:,4).*celldata.mds;
214         end
215         if isfield(header,'cell_data_modified')
216             celldata.perturbpds=header.cell_data_modified.PD(:,[1:ndim]);
217             celldata.perturbbs=[celldata.b0s,celldata.perturbpds];
218             celldata.perturbbs(:,2)=celldata.perturbbs(:,2).*celldata.mds;
219             celldata.perturbbs(:,3)=celldata.perturbbs(:,3).*celldata.mds;
220             if ndim==3
221                 celldata.perturbbs(:,4)=celldata.perturbbs(:,4).*celldata.mds;
222                 celldata.perturbaxis= repmat(header.bc.Axis,[length(celldata.mds),1]);
223             end
224             for i=1:size(celldata.pds,1)
225                 celldata.isperturbed(i,1)~=isequal(celldata.pds(i,:),celldata.perturbpds(i,:));
226             end
227         end
228         switch cellmode
229             case {'modulated'}
230                 headinds=find(header.cell_data.ModulatedCellsList);
231                 fns=fieldnames(celldata);
232                 for i=1:length(fns)
233                     celldata.(fns{i})=celldata.(fns{i})(headinds,:);
234                 end
235             case {'all'}
236                 headinds=[1:length(header.cell_data.ModulatedCellsList)];
237             otherwise
238                 error(sprintf('Unknown cellmode of %s',cellmode));
239         end
240         cellnamemat=cellstr(header.cell_data.Firing(headinds,:));
241         firstheadercd=header.cell_data;
242     else
243         if ~isequal(header.cell_data,firstheadercd)
244             errstr=sprintf('header cell data changed from file %s to file %s!',filenames{1},filenames{fi});
245             error(errstr);
246         end
247     end
248 end
249 end
250
251 %~~~~~
252
253 function [targset,ntrials]=get_sorted_targset_sf(app,filenames,ndim)
254 ntrials=0;
255 for fi=1:length(filenames)
256     load(filenames{fi});
257     if fi==1
258         targs=trials.TargetPos(:,[1:ndim]);
259     else
260         targs=[targs;trials.TargetPos(:,[1:ndim])];
261     end
262 end
263 targs=sc_normalize(app,targs);
264 ntrials=size(targs,1);
265 targset=unique(targs,'rows');
266 end
267 %~~~~~
268
269 function info=initialize_info(app,ntrials,ndim,ncells)
270
271 scalar_fieldlist={'targnum','seq_successful_trial_no','seq_trial_no'};
272 nby2vect_fieldlist={'spike_window'};
273 nbyncells_fieldlist={'rates'};
274
275 info.targnum=repmat(NaN,[ntrials,1]);
276 info.seq_successful_trial_no=repmat(NaN,[ntrials,1]);
277 info.seq_trial_no=repmat(NaN,[ntrials,1]);
278 info.spike_window=repmat(NaN,[ntrials,2]);
279 info.rates=repmat(NaN,[ntrials,ncells]);
280 end

```

```

281
282
283
284 function [spikewindow]=get_spike_window_sf(app, regressrange,j, trials,header, mvm, mvmntimes, ndim, synch, spk_window)
285
286 sw=nan(1,2); spikewindow=nan(1,2);
287 RT=150/1000*2;
288 short_window_width=200/1000;
289
290 switch regressrange
291     case {'all', 'half'}
292         sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j);
293     case {'all_rt', 'half_rt'}
294         sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+RT;
295     case {'half_short', 'short'}
296         if sw(2)-short_window_width>trials.ComputerTrialTime(j)+trials.HoldAFinish(j)
297             sw(1)=sw(2)-short_window_width;
298         end
299     case {'half_fixed'}
300         sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.3;
301     case {'half_fixeddelayed'}
302         sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.45;
303 end
304
305 switch regressrange
306     case {'all', 'all_rt'}
307         sw(2)=trials.HoldBStart(j)+trials.ComputerTrialTime(j);
308     case {'half', 'half_rt', 'half_short', 'short'}
309         sw(2) = sw(1) + spk_window;
310     case {'half_fixed'}
311         sw(2)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.5;
312     case {'half_fixeddelayed'}
313         sw(2)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.65;
314     otherwise
315         errstr=sprintf('I don''t know how to interpret regressrange %s',regressrange);
316         error('errstr');
317 end
318
319
320
321 if ~isnan(sw(2)) && ~isnan(sw(1)) && sw(2)>sw(1)
322     spikewindow=Computer2PlexonTime_sf(app,sw,header,synch);
323 end
324 end
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372 function y=sc_normalize(app,x)
373 nd=size(x,2);
374 nn=sqrt(sum(x.^2,2));
375 y=x./repmat(nn,[1,nd]);
376 return
377 end
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422

```

```

422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464

next_spiketime_after_last_bin = min(spike_times_wholetrial(find(spike_times_wholetrial > poi_stop)));
if isempty(next_spiketime_after_last_bin)
    next_spiketime_after_last_bin = max(spike_times_poi(end) + mean_ISI_poi, poi_stop);
end
spike_times = [last_spiketime_before_first_bin spike_times_poi next_spiketime_after_last_bin];

for bin_number = 1:length(start_times)
    start_time = start_times(bin_number);
    stop_time = stop_times(bin_number);
    spikes_in_bin = spike_times(spike_times > start_time & spike_times < stop_time);
    num_spikes_in_bin = length(spikes_in_bin);

    if num_spikes_in_bin == 0
        ISI_start = max(spike_times(find(spike_times <= start_times(bin_number))));
        ISI_end = min(spike_times(find(spike_times >= stop_times(bin_number))));
        ISI = ISI_end - ISI_start;
        part_int_count = binsize/ISI;
    elseif num_spikes_in_bin >= 1
        ISI_prev_start = max(spike_times(find(spike_times <= start_times(bin_number))));
        ISI_prev_end = spikes_in_bin(1);
        ISI_prev = ISI_prev_end - ISI_prev_start;
        first_fractional_count = (spikes_in_bin(1) - start_time)/ISI_prev;

        ISI_next_start = spikes_in_bin(end);
        ISI_next_end = min(spike_times(find(spike_times >= stop_times(bin_number))));
        ISI_next = ISI_next_end - ISI_next_start;
        last_fractional_count = (stop_time - spikes_in_bin(end))/ISI_next;

        additional_counts = num_spikes_in_bin - 1;
        part_int_count = first_fractional_count + additional_counts + last_fractional_count;
    end
    frs(bin_number) = part_int_count/binsize * 1000;
end
end
return
end

function fit_compute_cosine_fit(alltargs,allrates)
    [ntargs,ndim]=size(alltargs);
    [ntargs,ncells]=size(allrates);
    alltargs=[ones(ntargs,1),alltargs];

    targerr=inv(alltargs'*alltargs);
    for i=1:ncells
        % Now get the fits
        if any(allrates(:,i))
            [b,bsint,resid,rint,stats] = regress(allrates(:,i),alltargs,.95);
            fit.bs(i,:)=b;
            fit.bs(i,1)=b(1);
            fit.mds(i,1)=sqrt(sum(b(2:ndim+1).^2));
            fit.pds(i,:)=b(2:ndim+1)/fit.mds(i);
            fit.SSE(i,1)=sum(resid.^2);
            fit.RMSE(i,1)=sqrt(mean(resid.^2));
            fit.sigmas(i,1)=std(resid);
            if ndim==2
                [theta,phi]=cart2pol(b(2),b(3));
                fit.thetas(i,1)=theta*180/pi;
            else
                [theta,phi,phi]=cart2sph(b(2),b(3),b(4));
                fit.thetas(i,1)=theta*180/pi; fit.phis(i,1)=phi*180/pi;
            end
            fit.r2s(i,1)=stats(1);
            fit.pvals(i,1)=stats(3);
            fit.coefferr(:,i)=sum(resid.^2)/(length(resid)-(ndim+1)).*targerr;
            fit.islog(i,1)=false;
        else
            fit.bs(i,:)=NaN*ones(1,ndim+1);
            fit.bs(i,1)=NaN;
            fit.mds(i,1)=NaN;
            fit.pds(i,:)=NaN*ones(1,ndim);
            fit.SSE(i,1)=NaN;
            fit.RMSE(i,1)=NaN;
            fit.sigmas(i,1)=NaN;
            if ndim==2
                fit.thetas(i,1)=NaN;
            else
                fit.thetas(i,1)=NaN; fit.phis(i,1)=NaN;
            end
            fit.r2s(i,1)=NaN;
            fit.pvals(i,1)=NaN;
            fit.coefferr(:,i)=NaN*ones(ndim+1);
            fit.islog(i,1)=false;
        end
    end

    % alltargs=alltargs(:,[2:end]);
    % targs=unique(alltargs,'rows');
    % for j=1:size(targs,1)
    %     inds=find(alltargs(:,1)==targs(j,1) & alltargs(:,2)==targs(j,2));
    %     if ndim==3
    %         inds=intersect(inds,find(alltargs(:,3)==targs(j,3)));
    %     end
    %     if length(inds)>1
    %         fit.mnrates(j,:)=mean(allrates(inds,:));
    %     else
    %         fit.sdrates(j,:)=std(allrates(inds,:));
    %     end
    % fit.range=[min(fit.mnrates);max(fit.mnrates)];

    return
end
end

```

Startup function

```

538 % Callbacks that handle component events
539 methods (Access = private)
540
541 % Code that executes after component creation
542 function startupFcn(app, path)
543
544 % Set window name and only show in the day panel
545 % ChooseDayListBox and the buttons for selecting the days
546 app.TCWindow.Name = 'Tuning Curves';
547 app.SelectedDaysListBox.Visible = 'off';
548 app.SelectedDaysListBoxLabel.Visible = 'off';
549 app.DeleteDayButton.Visible = 'off';
550 app.DeleteAllDaysButton.Visible = 'off';
551 app.TargetCellPanel.Visible = 'off';
552 app.ChooseTargetCellListBox.Visible = 'off';
553 app.ChooseTargetCellListBoxLabel.Visible = 'off';
554 app.SelectTCButton.Visible = 'off';
555 app.SelectedTCTextArea.Visible = 'off';
556 app.SelectedTCTextAreaLabel.Visible = 'off';
557 app.NumTCAvailableTextArea.Visible = 'off';
558 app.NumTCAvailableTextAreaLabel.Visible = 'off';
559 app.PhasesPanel.Visible = 'off';
560 app.SelectedPhasesListBox.Visible = 'off';
561 app.SelectedPhasesListBoxLabel.Visible = 'off';
562 app.DeletePhaseButton.Visible = 'off';
563 app.DeleteAllPhasesButton.Visible = 'off';
564 app.SaveScreenshotButton.Visible = 'off';
565
566 app.selectedPath = path; % Save the path selected in the main window in app.selectedPath
567 a = dir(app.selectedPath); % Obtain the contents of the selected path.
568 b = {a(:).name}'; % Get the name of the folders of the contents and stores them appropriately in a cell array
569 b(ismember(b,{'.','..'})) = []; % Remove unnecessary '.' and '..' results from the display.
570
571 % Create a vector days with only the days whose files can be open
572 % without error
573 days = {};
574 for element = 1:length(b)
575     val = b{element};
576     error = 0;
577     try
578         path_file_inds = strcat(string(app.selectedPath), '\', string(val), '\file_inds.mat');
579         d = load(path_file_inds);
580         error = 0;
581     catch err
582         %errorDlg(err.message)
583         error = 1;
584     end
585     if error == 0
586         days(end+1) = b{element};
587     end
588
589 end
590
591 % Sort days in ascending order
592 dates = datetime(days, 'InputFormat','MM-dd-yy', 'Format','MM-dd-yy');
593 dates_ord = sort(dates);
594
595 % Stablish days shown in the ChooseDaysListBox
596 app.ChooseDaysListBox.Items = string(dates_ord);
597
598 % Stablish phases shown in ChoosePhaseListBox
599 app.phases = {'Control';'Perturbation';'Washout'};
600 app.ChoosePhasesListBox.Items = app.phases;
601
602 end

```

Callback function

```

631
632 % Button pushed function: SelectAllDaysButton
633 function SelectAllDaysButtonPushed(app, event)
634
635 % Stablish as SelectedDaysListbox items all the available days
636 % If it is empty, add all the days
637 if isempty(app.total_days)
638     app.total_days = [app.ChooseDaysListBox.Items];
639     app.SelectedDaysListBox.Items = app.total_days;
640
641 % If not, delete all items and add all the days to avoid
642 % repetitions
643 else
644     app.total_days = [];
645     app.total_days = [app.ChooseDaysListBox.Items];
646     app.SelectedDaysListBox.Items = app.total_days;
647 end
648
649 % Enable the user to visualize the next panel and the delete
650 % buttons
651 app.SelectedDaysListBox.Visible = 'on';
652 app.SelectedDaysListBoxLabel.Visible = 'on';
653 app.PhasesPanel.Visible = 'on';
654 app.DeleteDayButton.Visible = 'on';
655 app.DeleteAllDaysButton.Visible = 'on';
656
657 end
658
659 % Value changed function: SelectedDaysListBox
660 function SelectedDaysListBoxValueChanged(app, event)
661
662 % Stablish app.value_day as the item that is being selected
663 app.value_day = app.SelectedDaysListBox.Value;
664
665 end
666
667 % Button pushed function: DeleteDayButton
668 function DeleteDayButtonPushed(app, event)
669
670 % Delete the selected item by eliminating it from the vector
671 % using the index and update the SelectedDaysListBox
672 [~, index] = ismember(app.value_day, app.total_days);
673 app.total_days(index) = [];
674 app.SelectedDaysListBox.Items = app.total_days;
675
676 end
677
678 % Button pushed function: DeleteAllDaysButton
679 function DeleteAllDaysButtonPushed(app, event)
680
681 % Delete all days from the vector and all the items
682 app.total_days = [];
683 app.SelectedDaysListBox.Items = {};
684
685 end
686
687 % Value changed function: ChoosePhasesListBox
688 function ChoosePhasesListBoxValueChanged(app, event)
689
690 % Stablish app.phase_selected as the item selected and enable
691 % the user to choose more than one item
692 app.phase_selected = app.ChoosePhasesListBox.Value;
693 app.ChoosePhasesListBox.Multiselect = 'on';
694
695 end
696
697 % Button pushed function: SelectPhaseButton
698 function SelectPhaseButtonPushed(app, event)
699
700 % Avoid repeating a phase
701 if app.a == 0
702     app.a = app.a + 1;
703     add_phase = app.phase_selected;
704     app.phase_selected_total = [app.phase_selected_total; add_phase];
705
706 else
707     if ismember(app.phase_selected, app.phase_selected_total)
708     else
709         app.a = app.a + 1;
710         add_phase = app.phase_selected;
711         app.phase_selected_total = [app.phase_selected_total; add_phase];
712     end
713
714 end
715
716 % Enable the user to visualize the next panel and the delete
717 % buttons
718 app.SelectedPhasesListBox.Items = string(app.phase_selected_total);
719 app.SelectedPhasesListBox.Visible = 'on';
720 app.SelectedPhasesListBoxLabel.Visible = 'on';
721 app.TargetCellPanel.Visible = 'on';
722 app.DeleteAllPhasesButton.Visible = 'on';
723 app.DeletePhaseButton.Visible = 'on';
724
725 end

```

```

726
727 % Button pushed function: SelectAllPhasesButton
728 function SelectAllPhasesButtonPushed(app, event)
729
730 % Add all phases to the list considering if there are other
731 % items already selected
732 if isempty(app.phase_selected_total)
733     app.phase_selected_total = [app.phases];
734     app.SelectedPhasesListBox.Items = app.phase_selected_total;
735 else
736     app.phase_selected_total = [];
737     app.phase_selected_total = [app.phases];
738     app.SelectedPhasesListBox.Items = app.phase_selected_total;
739 end
740
741 % Stablish app.a to 3 and enable the visualization of the
742 % delete buttons and the last panel
743 app.a = 3;
744 app.SelectedPhasesListBox.Visible = 'on';
745 app.SelectedPhasesListBoxLabel.Visible = 'on';
746 app.TargetCellPanel.Visible = 'on';
747 app.DeleteAllPhasesButton.Visible = 'on';
748 app.DeletePhaseButton.Visible = 'on';
749 end
750
751 % Value changed function: SelectedPhasesListBox
752 function SelectedPhasesListBoxValueChanged(app, event)
753
754 % Stablish app.value to the phase selected in the
755 % SelectedPhaseListBox
756 app.value_phase = app.SelectedPhasesListBox.Value;
757
758 end
759
760 % Button pushed function: DeletePhaseButton
761 function DeletePhaseButtonPushed(app, event)
762
763 % Delete the phase chosen
764 [~, index] = ismember(app.value_phase, app.phase_selected_total);
765 app.phase_selected_total(index) = [];
766 app.SelectedPhasesListBox.Items = app.phase_selected_total;
767
768 % Reduce app.a as one phase has been deleted
769 app.a = app.a-1;
770
771 end
772
773 % Button pushed function: DeleteAllPhasesButton
774 function DeleteAllPhasesButtonPushed(app, event)
775
776 % Delete all phases selected
777 app.phase_selected_total = [];
778 app.SelectedPhasesListBox.Items = {};
779 app.a = 0; % Set app.a to 0
780
781 end
782
783 % Button pushed function: ShowTargetCellOptionsButton
784 function ShowTargetCellOptionsButtonPushed(app, event)
785
786 % Description of the variables
787 end_fname = '.CenterOut.mat';
788 days = app.total_days;
789 phase_sel = app.phase_selected_total;
790 app.C_control = {};
791 app.C_perturbation = {};
792 app.C_washout = {};
793 day_s = 1;
794 day_e = length(days);
795
796 % Extract the possible target cells for each day
797 for day_i = day_s:day_e
798     path_file_inds = strcat(string(app.selectedPath), '\', string(days(day_i)), 'file_inds.mat');
799     d = load(path_file_inds);
800     Const = fieldnames(d);
801
802     if app.a == 1
803
804         if strcmp('Control', phase_sel)
805
806             path_control = d.(Const{1});
807             path_length_control = length(path_control);
808
809             for file = 2:path_length_control
810                 try
811                     if isempty(path_control)
812                         else
813                             app.id_control = num2str(path_control(file));
814                             if numel(app.id_control) == 4
815                                 start_fname = 'Arthur.BC.0';
816                             else
817                                 start_fname = 'Arthur.BC.';
818                             end
819                             path_file_control = strcat(string(app.selectedPath), '\', string(days(day_i)), '\', start_fname, num2str(path_control(file)), end_fname);
820                             e_control = load(path_file_control);
821                             const1_control = fieldnames(e_control); % Variables that are included in this file
822                             struct_spikes_control = e_control.(const1_control{6}); % Spikes variable
823                             t_cells_control = fieldnames(struct_spikes_control); % List of all the units
824                             app.C_control{end+1} = t_cells_control(1:end-1);
825                             app.uvals_control = app.C_control{1};

```

```

826         % Keep oly the common target cells
827         for i_control = 1:numel(app.C_control)
828             app.uvals_control = intersect(app.C_control{i_control}, app.uvals_control); %Vector with the common target_cells
829         end
830     end
831     catch err
832         %errordlg(err.message)
833     end
834 end
835
836 elseif strcmp('Perturbation',phase_sel)
837     path_perturbation = d.(Const{2});
838     path_length_perturbation = length(path_perturbation);
839
840     for file = 2:path_length_perturbation
841         try
842             if isempty(path_perturbation)
843                 else
844                     app.id_perturbation = num2str(path_perturbation(file));
845                     if numel(app.id_perturbation) == 4
846                         start_fname = 'Arthur.BC.0';
847                     else
848                         start_fname = 'Arthur.BC.';
849                     end
850                     path_file_perturbation = strcat(string(app.selectedPath), '\\', string(days(day_i)), '\\', start_fname, num2str(path_perturbation(file)), end_fname);
851                     e_perturbation = load(path_file_perturbation);
852                     const1_perturbation = fieldnames(e_perturbation);
853                     struct_spikes_perturbation = e_perturbation.(const1_perturbation{6});
854                     t_cells_perturbation = fieldnames(struct_spikes_perturbation);
855                     app.C_perturbation(end+1) = t_cells_perturbation(1:end-1);
856                     app.uvals_perturbation = app.C_perturbation{1};
857                     for i_pert = 1:numel(app.C_perturbation)
858                         app.uvals_perturbation = intersect(app.C_perturbation{i_pert}, app.uvals_perturbation);
859                     end
860                 end
861             catch err
862                 %errordlg(err.message)
863             end
864         end
865     end
866
867 else
868     path_washout = d.(Const{3});
869     path_length_washout = length(path_washout);
870     for file = 2:path_length_washout
871         try
872             if isempty(path_washout)
873                 else
874                     app.id_washout = num2str(path_washout(file));
875                     if numel(app.id_washout) == 4
876                         start_fname = 'Arthur.BC.0';
877                     else
878                         start_fname = 'Arthur.BC.';
879                     end
880                     path_file_washout = strcat(string(app.selectedPath), '\\', string(days(day_i)), '\\', start_fname, num2str(path_washout(file)), end_fname);
881                     e_washout = load(path_file_washout);
882                     const1_washout = fieldnames(e_washout);
883                     struct_spikes_washout = e_washout.(const1_washout{6});
884                     t_cells_washout = fieldnames(struct_spikes_washout);
885                     app.C_washout(end+1) = t_cells_washout(1:end-1);
886                     app.uvals_washout = app.C_washout{1};
887                     for i_wash = 1:numel(app.C_washout)
888                         app.uvals_washout = intersect(app.C_washout{i_wash}, app.uvals_washout);
889                     end
890                 end
891             catch err
892                 %errordlg(err.message)
893             end
894         end
895     end
896 end
897
898 else
899     for element = 1:app.a
900         if ismember('Control', phase_sel(element))
901             path_control = d.(Const{1});
902             path_length_control = length(path_control);
903
904             for file = 2:path_length_control

```



```

907
908     try
909         if isempty(path_control)
910             else
911                 app.id_control = num2str(path_control(file));
912                 if numel(app.id_control) == 4
913                     start_fname = 'Arthur.BC.0';
914                 else
915                     start_fname = 'Arthur.BC.';
916                 end
917                 path_file_control = strcat(string(app.selectedPath), '\', string(days(day_i)), '\', start_fname, num2str(path_control(file)), end_fname);
918                 e_control = load(path_file_control);
919                 const1_control = fieldnames(e_control);
920                 struct_spikes_control = e_control.(const1_control{6});
921                 t_cells_control = fieldnames(struct_spikes_control);
922                 app.C_control(end+1) = t_cells_control(1:end-1);
923                 app.uvals_control = app.C_control{1};
924                 for i_control = 1:numel(app.C_control)
925                     app.uvals_control = intersect(app.C_control{i_control}, app.uvals_control);
926                 end
927             end
928         catch err
929             %errordlg(err.message)
930         end
931     end
932
933     elseif strcmp('Perturbation', phase_sel(element))
934         path_perturbation = d.(Const{2});
935         path_length_perturbation = length(path_perturbation);
936
937         for file = 2:path_length_perturbation
938             try
939                 if isempty(path_perturbation)
940                     else
941                         app.id_perturbation = num2str(path_perturbation(file));
942                         if numel(app.id_perturbation) == 4
943                             start_fname = 'Arthur.BC.0';
944                         else
945                             start_fname = 'Arthur.BC.';
946                         end
947
948                         path_file_perturbation = strcat(string(app.selectedPath), '\', string(days(day_i)), '\', start_fname, num2str(path_perturbation(file)), end_fname);
949                         e_perturbation = load(path_file_perturbation);
950                         const1_perturbation = fieldnames(e_perturbation);
951                         struct_spikes_perturbation = e_perturbation.(const1_perturbation{6});
952                         t_cells_perturbation = fieldnames(struct_spikes_perturbation);
953                         app.C_perturbation(end+1) = t_cells_perturbation(1:end-1);
954                         app.uvals_perturbation = app.C_perturbation{1};
955                         for i_pert = 1:numel(app.C_perturbation)
956                             app.uvals_perturbation = intersect(app.C_perturbation{i_pert}, app.uvals_perturbation);
957                         end
958                     catch err
959                         %errordlg(err.message)
960                     end
961                 end
962             end
963         else
964             path_washout = d.(Const{3});
965             path_length_washout = length(path_washout);
966             for file = 2:path_length_washout
967                 try
968                     if isempty(path_washout)
969                         else
970                             app.id_washout = num2str(path_washout(file));
971                             if numel(app.id_washout) == 4
972                                 start_fname = 'Arthur.BC.0';
973                             else
974                                 start_fname = 'Arthur.BC.';
975                             end
976                             path_file_washout = strcat(string(app.selectedPath), '\', string(days(day_i)), '\', start_fname, num2str(path_washout(file)), end_fname);
977                             e_washout = load(path_file_washout);
978                             const1_washout = fieldnames(e_washout);
979                             struct_spikes_washout = e_washout.(const1_washout{6});
980                             t_cells_washout = fieldnames(struct_spikes_washout);
981                             app.C_washout(end+1) = t_cells_washout(1:end-1);
982                             app.uvals_washout = app.C_washout{1};
983                             for i_wash = 1:numel(app.C_washout)
984                                 app.uvals_washout = intersect(app.C_washout{i_wash}, app.uvals_washout);
985                             end
986                         end
987                     catch err
988                         %errordlg(err.message)
989                     end
990                 end
991             end

```

```

992         end
993     end
994     end
995 end
996
997
998 % Obtain the final list of target cells. Only the ones ending
999 % in 1 can be used.
1000 if app.a == 1
1001     if strcmp('Control', phase_sel)
1002         app.target_cells_list = [];
1003         for i = 1:length(app.uvals_control)
1004             item = char(app.uvals_control(i));
1005             if str2double(item(end)) == 1
1006                 app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1007             else
1008                 end
1009             end
1010         end
1011
1012     elseif strcmp('Perturbation', phase_sel )
1013
1014         app.target_cells_list = [];
1015         for i = 1:length(app.uvals_perturbation)
1016             item = char(app.uvals_perturbation(i));
1017             if str2double(item(end)) == 1
1018                 app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1019             else
1020                 end
1021             end
1022         end
1023     else
1024         app.target_cells_list = [];
1025         for i = 1:length(app.uvals_washout)
1026             item = char(app.uvals_washout(i));
1027             if str2double(item(end)) == 1
1028                 app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1029             else
1030                 end
1031             end
1032         end
1033     end
1034
1035 %If more than one phase is selected, the common target cells
1036 %for all the phases need to be extracted
1037 elseif app.a == 2
1038     if ismember('Control', phase_sel(1)) || ismember('Control', phase_sel(2))
1039         if ismember('Perturbation', phase_sel(1)) || ismember('Perturbation', phase_sel(2))
1040             u = app.uvals_control;
1041             u1 = {};
1042             for i = 1:numel(app.uvals_perturbation)
1043                 u2 = char(intersect(app.uvals_perturbation(i)', u));
1044                 if isempty(u2)
1045                     else
1046                         u1{end+1} = u2;
1047                     end
1048                 end
1049             end
1050             app.target_cells_list = [];
1051             for i = 1:length(u1)
1052                 item = char(u1(i));
1053                 if str2double(item(end)) == 1
1054                     app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1055                 else
1056                     end
1057             end
1058         end
1059     else
1060         u = app.uvals_control;
1061         u1 = {};
1062         for i = 1:numel(app.uvals_washout)
1063             u2 = char(intersect(app.uvals_washout(i)', u));
1064             if isempty(u2)
1065                 else
1066                     u1{end+1} = u2;
1067                 end
1068             end
1069         end
1070         app.target_cells_list = [];
1071         for i = 1:length(u1)
1072             item = char(u1(i));
1073             if str2double(item(end)) == 1
1074                 app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1075             else
1076                 end
1077             end
1078         end
1079     end
1080 end
1081 else

```

```

1075
1076 -         u = app.uvals_perturbation;
1077 -         u1 = {};
1078 -         for i = 1:numel(app.uvals_washout)
1079 -             u2 = char(intersect(app.uvals_washout(i)', u));
1080 -             if isempty(u2)
1081 -                 else
1082 -                     u1[end+1] = u2;
1083 -                 end
1084 -             end
1085 -             app.target_cells_list = [];
1086 -             for i = 1:length(u1)
1087 -                 item = char(u1(i));
1088 -                 if str2double(item(end)) == 1
1089 -                     app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1090 -                 else
1091 -                     end
1092 -                 end
1093 -             end
1094 -         else
1095 -             u = app.uvals_control;
1096 -             u1 = {};
1097 -             for i = 1:numel(app.uvals_perturbation)
1098 -                 u2 = char(intersect(app.uvals_perturbation(i)', u));
1099 -                 if isempty(u2)
1100 -                     else
1101 -                         u1[end+1] = u2;
1102 -                     end
1103 -                 end
1104 -                 uvals_def = {};
1105 -                 for i = 1:numel(app.uvals_washout)
1106 -                     u3 = char(intersect(app.uvals_washout(i)', u1));
1107 -                     if isempty(u3)
1108 -                         else
1109 -                             uvals_def[end+1] = u3;
1110 -                         end
1111 -                     end
1112 -                 end
1113 -                 app.target_cells_list = [];
1114 -                 for i = 1:length(uvals_def)
1115 -                     item = char(uvals_def(i));
1116 -                     if str2double(item(end)) == 1
1117 -                         app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
1118 -                     else
1119 -                         end
1120 -                     end
1121 -                 end
1122 -             end
1123 -             % Use the definitive target cells list as items for the
1124 -             % ChooseTargetCellListBox and enable the visualization of the
1125 -             % SelectTCButton and the number of TC available
1126 -             app.ChooseTargetCellListBox.Items = app.target_cells_list;
1127 -             app.ChooseTargetCellListBox.Visible = 'on';
1128 -             app.ChooseTargetCellListBoxLabel.Visible = 'on';
1129 -             app.SelectTCButton.Visible = 'on';
1130 -             app.NumTCAvailableTextArea.Visible = 'on';
1131 -             app.NumTCAvailableTextAreaLabel.Visible = 'on';
1132 -             app.NumTCAvailableTextArea.Value = string(length(app.target_cells_list));
1133 -         end
1134 -     end
1135 -
1136 - % Value changed function: ChooseTargetCellListBox
1137 - function ChooseTargetCellListBoxValueChanged(app, event)
1138 -
1139 -     % Stablish app.target_cell as the selected target cell and
1140 -     % enable the Multiselect option
1141 -     app.target_cell = app.ChooseTargetCellListBox.Value;
1142 -     app.ChooseTargetCellListBox.Multiselect = 'off';
1143 - end
1144 -

```

```

1146 % Button pushed function: SelectTCButton
1147 function SelectTCButtonPushed(app, event)
1148
1149 % Show the selected TC
1150 app.SelectedTCTextArea.Visible = 'on';
1151 app.SelectedTCTextAreaLabel.Visible = 'on';
1152 app.SaveScreenshotButton.Visible = 'on';
1153 app.SelectedTCTextArea.Value = app.target_cell;
1154
1155 % Description of the variables
1156 app.all_sentences_control = [];
1157 app.all_sentences_perturbation = [];
1158 app.all_sentences_washout = [];
1159 path = app.selectedPath;
1160 days = app.total_days;
1161 phase_sel = app.phase_selected_total;
1162 end_fname = '.CenterOut.mat';
1163 target_cell = app.target_cell;
1164 day_s = 1;
1165 day_e = length(days);
1166 num_days = day_e - day_s + 1;
1167 gray_panel = uipanel(app.TCWindow, 'Position', [0, 0, 1536, 600], 'Scrollable', 'on'); % To cover the existing curves if that is the case
1168
1169 % Obtain and plot the tuning curves
1170 if app.a == 1
1171     panel = uipanel(app.TCWindow, 'Position', [100, 40, 1336, 560], 'Scrollable', 'on');
1172
1173     for day_i = day_s:day_e
1174         app.all_rates_control = [];
1175         app.all_rates_perturb = [];
1176         app.all_rates_washout = [];
1177         path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
1178         d = load(path_file_inds);
1179         Const = fieldnames(d);
1180         path_control = d.(Const{1}); % Identification number of the control experiment files
1181         path_length_control = length(path_control); % Number of control experiment files
1182         path_perturbation = d.(Const{2}); % Identification number of the perturbation experiment files
1183         path_length_perturbation = length(path_perturbation); % Number of perturbation experiment files
1184         path_washout = d.(Const{3}); % Identification number of the washout experiment files
1185         path_length_washout = length(path_washout); % Number of washout experiment files
1186
1187     if strcmp('Control', phase_sel)
1188         % Create a panel and establish its characteristics
1189         panel.Title = 'Control';
1190         panel.FontSize = 16;
1191         panel.Fontweight = 'Bold';
1192
1193         for file = 2:path_length_control
1194             try
1195                 app.rates_control = [];
1196                 ang = [];
1197                 i=1;
1198                 if isempty(path_control)
1199                     else
1200                         app.id_control = num2str(path_control(file));
1201                         if numel(app.id_control) == 4
1202                             start_fname = 'Arthur.BC.0';
1203                         else
1204                             start_fname = 'Arthur.BC.';
1205                         end
1206                         app.path_file_control = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_control(file)), end_fname);
1207                         e_control = load(app.path_file_control);
1208                         [info_control, target_control, app.cellnames_control, celldata_control] = get_seq_rate_info_short_win(app, app.path_file_control, 'half_rt', 'rate', 0.3);
1209
1210                         % Determine the index of the target
1211                         % cell selected in the list pf cellnames
1212                         app.cell_ind_control = [];
1213                         for cname = 1:length(app.cellnames_control)
1214                             cell_val_control = app.cellnames_control{cname};
1215                             if cell_val_control == target_cell
1216                                 app.cell_ind_control = cname;
1217                             end
1218                         end
1219                         if isempty(app.cell_ind_control)
1220                             app.cell_ind_control = 1;
1221                         end
1222
1223                         % Obtain the angles and the rates
1224                         for cell_control = app.cell_ind_control

```

```

1225
1226 -         targets_control = e_control.trials.TargetPos(:,1:2);
1227 -         angles_rad_control = atan(targets_control(:,2)./targets_control(:,1));
1228 -         ang_control = angles_rad_control * 360 / (2*pi);
1229 -         ind1 = find(targets_control(:,1)<0 & targets_control(:,2)>0);
1230 -         ang_control(ind1) = ang_control(ind1)+180;
1231 -         ind2 = find(targets_control(:,1)<0 & targets_control(:,2)<0);
1232 -         ang_control(ind2) = ang_control(ind2)-180;
1233 -         ind3 = find(targets_control(:,1)<0 & targets_control(:,2)==0);
1234 -         ang_control(ind3) = 180;
1235 -         [list, ind_control] = sort(ang_control);
1236
1237 -         ind_sort_targ_control = ind_control;
1238 -         targets_sort_control = targets_control(ind_sort_targ_control,:);
1239 -         angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
1240 -         app.angle_control = angle_rad_control * 360 / (2*pi);
1241 -         ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
1242 -         app.angle_control(ind1) = app.angle_control(ind1)+180;
1243 -         ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
1244 -         app.angle_control(ind2) = app.angle_control(ind2)-180;
1245 -         ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
1246 -         app.angle_control(ind3) = 180;
1247
1248 -         sort_rates_control = info_control.rates(ind_sort_targ_control,:);
1249 -         sr_cell_control = sort_rates_control(:,cell_control);
1250 -         app.rates_control(:,1) = sr_cell_control';
1251
1252 -         early_control_fit = compute_cosine_fit(targets_control,app.rates_control); % Obtain the parameters needed to fit cosine tuning curves
1253
1254 -     end
1255
1256 -     end
1257 -     % Include all the rates in a single matrix
1258 -     app.all_rates_control = [app.all_rates_control sr_cell_control];
1259 - catch err
1260 -     %errorldg(err.message)
1261 - end
1262 - end
1263
1264 - try
1265 -     % Calculate the mean of all the rates, to
1266 -     % obtain one rate per angle
1267 -     mean_all_rates_control = mean(app.all_rates_control');
1268 -     % Calculate the maximum value of the mean
1269 -     % vector and to which angle does it correspond
1270 -     [max_value_control, index_max_value_control] = max(mean_all_rates_control');
1271 -     angle_max_control = app.angle_control(index_max_value_control);
1272 -     % Generate a sentence with the angle of
1273 -     % preference and its spiking rate
1274 -     sentence_control = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_control, max_value_control);
1275 -     app.all_sentences_control = [app.all_sentences_control, sentence_control];
1276
1277 -     % Generate the plot rates vs angles, plot a
1278 -     % line with the values of the means and draw
1279 -     % a dot to indicate the maximum rate (that
1280 -     % corresponds to the angle of preference)
1281 -     ax_control = uiaxes(panel,'Position',[40, 140*(num_days-day_i)+10,1256,130]);
1282 -     hold(ax_control, 'on')
1283 -     plot(ax_control, app.angle_control,app.all_rates_control,'b','MarkerSize',12)
1284 -     plot(ax_control,app.angle_control,mean_all_rates_control, 'LineWidth',4)
1285 -     plot(ax_control, angle_max_control,max_value_control,'r','MarkerSize', 12)
1286 -     xlabel(ax_control, 'Angle','FontSize',12)
1287 -     ylabel(ax_control, 'Spiking rate','FontSize',12)
1288 -     title(ax_control, strcat('Day: ', days(day_i)))
1289 -     hold(ax_control, "off")
1290
1291 - catch err
1292 -     %errorldg(err.message)
1293 - end
1294
1295 - elseif strcmp('Perturbation',phase_sel)
1296
1297 -     panel.Title = 'Perturbation';
1298 -     panel.FontSize = 16;
1299 -     panel.FontWeight = 'Bold';
1300 -     for file = 2:path_length_perturbation
1301 -         try
1302 -             app.rates_perturb = [];
1303 -             ang = [];
1304 -             i=1;
1305 -             if isempty(path_perturbation)

```

```

1306 -     else
1307 -         app.id_perturbation = num2str(path_perturbation(file));
1308 -         if numel(app.id_perturbation) == 4
1309 -             start_fname = 'Arthur.BC.0';
1310 -         else
1311 -             start_fname = 'Arthur.BC.';
1312 -         end
1313 -         app.path_file_perturb = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_perturbation(file)), end_fname);
1314 -         e_perturb = load(app.path_file_perturb);
1315 -         [info_perturb, targetset_perturb, app.cellnames_perturb, cellldata_perturb] = get_seq_rate_info_short_win(app, app.path_file_perturb, 'half_rt', 'rate', 0.3);
1316 -
1317 -         app.cell_ind_perturb = [];
1318 -         for cname = 1:length(app.cellnames_perturb) %cellnames_valid
1319 -             cell_val_perturb = app.cellnames_perturb(cname); % coge el numero de unidad --> 18_1
1320 -             if cell_val_perturb == target_cell
1321 -                 app.cell_ind_perturb = cname;
1322 -             end
1323 -         end
1324 -
1325 -         if isempty(app.cell_ind_perturb)
1326 -             app.cell_ind_perturb = 1;
1327 -         end
1328 -         for cell_perturb = app.cell_ind_perturb
1329 -             targets_perturb = e_perturb.trials.TargetPos(:,1:2);
1330 -             angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
1331 -             ang_perturb = angles_rad_perturb * 360 / (2*pi);
1332 -             ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
1333 -             ang_perturb(ind1) = ang_perturb(ind1)+180;
1334 -             ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
1335 -             ang_perturb(ind2) = ang_perturb(ind2)-180;
1336 -             ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
1337 -             ang_perturb(ind3) = 180;
1338 -
1339 -             [list, ind_perturb] = sort(ang_perturb);
1340 -             ind_sort_targ_perturb = ind_perturb;
1341 -             targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);
1342 -             angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
1343 -             app.angle_perturb = angle_rad_perturb * 360 / (2*pi);
1344 -             ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
1345 -             app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
1346 -             ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
1347 -             app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
1348 -             ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
1349 -             app.angle_perturb(ind3) = 180;
1350 -
1351 -             sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
1352 -             sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
1353 -             app.rates_perturb(:,i) = sr_cell_perturb';
1354 -
1355 -             early_perturb_fit = compute_cosine_fit(targets_perturb, app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
1356 -         end
1357 -     end
1358 -     app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
1359 - catch err
1360 -     %errordlg(err.message)
1361 - end
1362 - end
1363 - try
1364 -     mean_all_rates_perturb = mean(app.all_rates_perturb');
1365 -     [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
1366 -     angle_max_perturb = app.angle_perturb(index_max_value_perturb);
1367 -     sentence_perturb = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_perturb, max_value_perturb);
1368 -     app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
1369 -     ax_perturb = uiaxes(panel, 'Position', [40, 140*(num_days-day_i)+10, 1256, 130]);
1370 -     hold(ax_perturb, 'on')
1371 -     plot(ax_perturb, app.angle_perturb, app.all_rates_perturb, 'b.', 'MarkerSize', 12)
1372 -     plot(ax_perturb, app.angle_perturb, mean_all_rates_perturb, 'LineWidth', 4)
1373 -     plot(ax_perturb, angle_max_perturb, max_value_perturb, 'r.', 'MarkerSize', 12)
1374 -     xlabel(ax_perturb, 'Angle', 'FontSize', 12)
1375 -     ylabel(ax_perturb, 'Spiking rate', 'FontSize', 12)
1376 -     title(ax_perturb, strcat('Day: ', days(day_i)))
1377 -     hold(ax_perturb, "off")
1378 - catch err
1379 -     %errordlg(err.message)
1380 - end
1381 - else
1382 -     app.ChooseDaysListBox.Items = string(phase_sel);
1383 -     panel.Title = 'Washout';
1384 -     panel.FontSize = 16;
1385 -     panel.FontWeight = 'Bold';
1386 -     for file = 2:path_length_washout
1387 -         try
1388 -             app.rates_washout = [];
1389 -             ang = [];
1390 -             i=1;
1391 -             if isempty(path_washout)
1392 -                 else

```

```

1393 - app.id_washout = num2str(path_washout(file));
1394 - if numel(app.id_washout) == 4
1395 -     start_fname = 'Arthur.BC.0';
1396 - else
1397 -     start_fname = 'Arthur.BC.';
1398 - end
1399 - app.path_file_washout = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_washout(file)), end_fname);
1400 - e_washout = load(app.path_file_washout);
1401 - [info_washout, targetset_washout, app.cellnames_washout, celldata_washout] = get_seq_rate_info_short_win(app, app.path_file_washout, 'half_rt', 'rate', 0.3);
1402 -
1403 - app.cell_ind_washout = [];
1404 - for cname = 1:length(app.cellnames_washout)
1405 -     cell_val_washout = app.cellnames_washout(cname);
1406 -     if cell_val_washout == target_cell
1407 -         app.cell_ind_washout = cname;
1408 -     end
1409 - end
1410 - if isempty(app.cell_ind_washout)
1411 -     app.cell_ind_washout = 1;
1412 - end
1413 - for cell_washout = app.cell_ind_washout
1414 -     targets_washout = e_washout.trials.TargetPos(:,1:2);
1415 -     angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
1416 -     ang_washout = angles_rad_washout * 360 / (2*pi);
1417 -     ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
1418 -     ang_washout(ind1) = ang_washout(ind1)+180;
1419 -     ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
1420 -     ang_washout(ind2) = ang_washout(ind2)-180;
1421 -     ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
1422 -     ang_washout(ind3) = 180;
1423 -     [list, ind_washout] = sort(ang_washout);
1424 -     ind_sort_targ_washout = ind_washout;
1425 -     targets_sort_washout = targets_washout(ind_sort_targ_washout,:);
1426 -
1427 -     angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
1428 -     app.angle_washout = angle_rad_washout * 360 / (2*pi);
1429 -     ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
1430 -     app.angle_washout(ind1) = app.angle_washout(ind1)+180;
1431 -     ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
1432 -     app.angle_washout(ind2) = app.angle_washout(ind2)-180;
1433 -     ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
1434 -     app.angle_washout(ind3) = 180;
1435 -
1436 -     sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
1437 -     sr_cell_washout = sort_rates_washout(:,cell_washout);
1438 -     app.rates_washout(:,i) = sr_cell_washout;
1439 -
1440 -     early_washout_fit = compute_cosine_fit(targets_washout, app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
1441 - end
1442 - end
1443 - app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
1444 - catch err
1445 -     %errordlg(err.message)
1446 - end
1447 - end
1448 - try
1449 -     mean_all_rates_washout = mean(app.all_rates_washout);
1450 -     [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout);
1451 -     angle_max_washout = app.angle_washout(index_max_value_washout);
1452 -     sentence_washout = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_washout, max_value_washout);
1453 -     app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
1454 -     ax_washout = uiaxes(panel, 'Position', [40, 140*(num_days-day_i)+10, 1256, 130]);
1455 -     hold(ax_washout, 'on')
1456 -     plot(ax_washout, app.angle_washout, app.all_rates_washout, 'b.', 'MarkerSize', 12)
1457 -     plot(ax_washout, app.angle_washout, mean_all_rates_washout, 'LineWidth', 4)
1458 -     plot(ax_washout, angle_max_washout, max_value_washout, 'r.', 'MarkerSize', 12)
1459 -     xlabel(ax_washout, 'Angle', 'FontSize', 12)
1460 -     ylabel(ax_washout, 'Spiking rate', 'FontSize', 12)
1461 -     title(ax_washout, strcat('Day: ', days(day_i)))
1462 -     hold(ax_washout, 'off')
1463 - catch err
1464 -     %errordlg(err.message)
1465 - end
1466 - end

```

```

1467     end
1468 elseif app.a == 2
1469     panel1 = uipanel(app.TCWindow, 'Position', [40, 40, 708, 560], 'Scrollable', 'on' );
1470     panel2 = uipanel(app.TCWindow, 'Position', [788, 40, 708, 560], 'Scrollable', 'on' );
1471     for day_i = day_s:day_e
1472         app.all_rates_control = [];
1473         app.all_rates_perturb = [];
1474         app.all_rates_washout = [];
1475         path_file_innds = strcat(string(path), '\', string(days(day_i)), '\file_innds.mat');
1476         d = load(path_file_innds);
1477         Const = fieldnames(d);
1478         path_control = d.(Const{1});
1479         path_length_control = length(path_control);
1480         path_perturbation = d.(Const{2});
1481         path_length_perturbation = length(path_perturbation);
1482         path_washout = d.(Const{3});
1483         path_length_washout = length(path_washout);
1484         if ismember('Control', phase_sel(1)) || ismember('Control', phase_sel(2))
1485             panel1.Title = 'Control';
1486             panel1.FontSize = 16;
1487             panel1.FontWeight = 'Bold';
1488             for file = 2:path_length_control
1489                 try
1490                     app.rates_control = [];
1491                     ang = [];
1492                     i=1;
1493                     if isempty(path_control)
1494                         else
1495                             app.id_control = num2str(path_control(file));
1496                             if numel(app.id_control) == 4
1497                                 start_fname = 'Arthur.BC.0';
1498                             else
1499                                 start_fname = 'Arthur.BC.';
1500                             end
1501                             app.path_file_control = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_control(file)), end_fname);
1502                             e_control = load(app.path_file_control);
1503                             [info_control, target_control, app.cellnames_control, celldata_control] = get_seq_rate_info_short_win(app, app.path_file_control,
1504                                 ['half_rt', 'rate', 0.3]);
1505
1506                             app.cell_ind_control = [];
1507                             for cname = 1:length(app.cellnames_control)
1508                                 cell_val_control = app.cellnames_control{cname};
1509                                 if cell_val_control == target_cell
1510                                     app.cell_ind_control = cname;
1511                                 end
1512                             end
1513
1514                             if isempty(app.cell_ind_control)
1515                                 app.cell_ind_control = 1;
1516                             end
1517                             for cell_control = app.cell_ind_control
1518                                 targets_control = e_control.trials.TargetPos(:,1:2);
1519                                 angles_rad_control = atan(targets_control(:,2)./targets_control(:,1));
1520                                 ang_control = angles_rad_control * 360 / (2*pi);
1521                                 ind1 = find(targets_control(:,1)<0 & targets_control(:,2)>0);
1522                                 ang_control(ind1) = ang_control(ind1)+180;
1523                                 ind2 = find(targets_control(:,1)<0 & targets_control(:,2)<0);
1524                                 ang_control(ind2) = ang_control(ind2)-180;
1525                                 ind3 = find(targets_control(:,1)<0 & targets_control(:,2)==0);
1526                                 ang_control(ind3) = 180;
1527                                 [list, ind_control] = sort(ang_control);
1528                                 ind_sort_targ_control = ind_control;
1529                                 targets_sort_control = targets_control(ind_sort_targ_control,:);
1530
1531                                 angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
1532                                 app.angle_control = angle_rad_control * 360 / (2*pi);
1533                                 ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
1534                                 app.angle_control(ind1) = app.angle_control(ind1)+180;
1535                                 ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
1536                                 app.angle_control(ind2) = app.angle_control(ind2)-180;
1537                                 ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
1538                                 app.angle_control(ind3) = 180;
1539
1540                                 sort_rates_control = info_control.rates(ind_sort_targ_control,:);
1541                                 sr_cell_control = sort_rates_control(:,cell_control);
1542                                 app.rates_control(:,i) = sr_cell_control';
1543
1544                                 early_control_fit = compute_cosine_fit(targets_control, app.rates_control); % Obtain the parameters needed to fit cosine tuning curves
1545                             end
1546                             app.all_rates_control = [app.all_rates_control sr_cell_control];
1547                         catch err
1548                             %erroridg(err.message)
1549                         end
1550                     end
1551                 end
1552             end
1553         end
1554     end
1555 end

```



```

1551 - try
1552 -     mean_all_rates_control = mean(app.all_rates_control');
1553 -     [max_value_control, index_max_value_control] = max(mean_all_rates_control');
1554 -     angle_max_control = app.angle_control(index_max_value_control);
1555 -     sentence_control = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_control, ...
1556 -         max_value_control);
1557 -     app.all_sentences_control = [app.all_sentences_control, sentence_control];
1558 -     ax_control = uiaxes(panell1, 'Position', [10, 140*(num_days-day_i)+10, 688, 130]);
1559 -     hold(ax_control, 'on')
1560 -     plot(ax_control, app.angle_control, app.all_rates_control, 'b.', 'MarkerSize', 12)
1561 -     plot(ax_control, app.angle_control, mean_all_rates_control, 'LineWidth', 4)
1562 -     plot(ax_control, angle_max_control, max_value_control, 'r.', 'MarkerSize', 12)
1563 -     xlabel(ax_control, 'Angle', 'FontSize', 12)
1564 -     ylabel(ax_control, 'Spiking rate', 'FontSize', 12)
1565 -     title(ax_control, strcat('Day: ', days(day_i)))
1566 -     hold(ax_control, 'off')
1567 - catch err
1568 -     %errorldg(err.message)ME
1569 - end
1570 - if ismember('Perturbation', phase_sel(1)) || ismember('Perturbation', phase_sel(2))
1571 -     panel2.Title = 'Perturbation';
1572 -     panel2.FontSize = 16;
1573 -     panel2.FontWeight = 'Bold';
1574 -     for file = 2:path_length_perturbation
1575 -         try
1576 -             app.rates_perturb = [];
1577 -             ang = [];
1578 -             i=1;
1579 -             if isempty(path_perturbation)
1580 -                 else
1581 -                     app.id_perturbation = num2str(path_perturbation(file));
1582 -                     if numel(app.id_perturbation) == 4
1583 -                         start_fname = 'Arthur.BC.0';
1584 -                     else
1585 -                         start_fname = 'Arthur.BC.';
1586 -                     end
1587 -
1588 - app.path_file_perturb = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_perturbation(file)), end_fname);
1589 - e_perturb = load(app.path_file_perturb);
1590 - [info_perturb, targetset_perturb, app.cellnames_perturb, celldata_perturb] = get_seq_rate_info_short_win(app, app.path_file_perturb, 'half_rt', 'rate', 0.3);
1591 - app.cellnames_valid_perturb = [];
1592 -
1593 - app.cell_ind_perturb = [];
1594 - for cname = 1:length(app.cellnames_perturb)
1595 -     cell_val_perturb = app.cellnames_perturb(cname);
1596 -     if cell_val_perturb == target_cell
1597 -         app.cell_ind_perturb = cname;
1598 -     end
1599 - end
1600 - if isempty(app.cell_ind_perturb)
1601 -     app.cell_ind_perturb = 1;
1602 - end
1603 - for cell_perturb = app.cell_ind_perturb
1604 -     targets_perturb = e_perturb.trials.TargetPos(:,1:2);
1605 -     angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
1606 -     ang_perturb = angles_rad_perturb * 360 / (2*pi);
1607 -     ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
1608 -     ang_perturb(ind1) = ang_perturb(ind1)+180;
1609 -     ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
1610 -     ang_perturb(ind2) = ang_perturb(ind2)-180;
1611 -     ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
1612 -     ang_perturb(ind3) = 180;
1613 -     [list, ind_perturb] = sort(ang_perturb);
1614 -     ind_sort_targ_perturb = ind_perturb;
1615 -     targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);
1616 -
1617 -     angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
1618 -     app.angle_perturb = angle_rad_perturb * 360 / (2*pi);
1619 -     ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
1620 -     app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
1621 -     ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
1622 -     app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
1623 -     ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
1624 -     app.angle_perturb(ind3) = 180;
1625 -
1626 -     sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
1627 -     sr_cell_perturb = sort_rates_perturb(:,ind_perturb);
1628 -     app.rates_perturb(:,i) = sr_cell_perturb;
1629 -
1630 -     early_perturb_fit = compute_cosine_fit(targets_perturb, app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
1631 - end

```

```

1632 -         end
1633 -         app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
1634 -     catch err
1635 -         %rerrordlg(err.message)
1636 -     end
1637 - end
1638 - try
1639 -
1640 -     mean_all_rates_perturb = mean(app.all_rates_perturb');
1641 -     [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
1642 -     angle_max_perturb = app.angle_perturb(index_max_value_perturb);
1643 -     sentence_perturb = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_perturb, ...
1644 -         max_value_perturb);
1645 -     app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
1646 -     ax_perturb = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,688,130]);
1647 -     hold(ax_perturb, 'on')
1648 -     plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
1649 -     plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
1650 -     plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
1651 -     xlabel(ax_perturb,'Angle','FontSize',12)
1652 -     ylabel(ax_perturb, 'Spiking rate','FontSize',12)
1653 -     title(ax_perturb, strcat('Day: ', days(day_i)))
1654 -     hold(ax_perturb,"off")
1655 - catch err
1656 -     %rerrordlg(err.message)
1657 - end
1658 - else
1659 -     panel2.Title = 'Washout';
1660 -     panel2.FontSize = 16;
1661 -     panel2.FontWeight = 'Bold';
1662 -     for file = 2:path_length_washout
1663 -         try
1664 -             app.rates_washout = [];
1665 -             ang = [];
1666 -             i=1;
1667 -             if isempty(path_washout)
1668 -                 else
1669 -                     app.id_washout = num2str(path_washout(file));
1670 -                     if numel(app.id_washout) == 4
1671 -                         start_fname = 'Arthur.BC.0';
1672 -                     else
1673 -                         start_fname = 'Arthur.BC.';
1674 -                     end
1675 -
1676 -             app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
1677 -             e_washout = load(app.path_file_washout);
1678 -             [info_washout,targetset_washout,app.cellnames_washout,celldata_washout]= get_seq_rate_info_short_win(app, app.path_file_washout, ...
1679 -                 ['half_rt','rate',0.3]);
1680 -
1681 -             app.cell_ind_washout = [];
1682 -             for cname = 1:length(app.cellnames_washout)
1683 -                 cell_val_washout = app.cellnames_washout{cname};
1684 -                 if cell_val_washout == target_cell
1685 -                     app.cell_ind_washout = cname;
1686 -                 end
1687 -             end
1688 -             if isempty(app.cell_ind_washout)
1689 -                 app.cell_ind_washout = 1;
1690 -             end
1691 -             for cell_washout = app.cell_ind_washout
1692 -                 targets_washout = e_washout.trials.TargetPos(:,1:2);
1693 -                 angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
1694 -                 ang_washout = angles_rad_washout * 360 / (2*pi);
1695 -                 ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
1696 -                 ang_washout(ind1) = ang_washout(ind1)+180;
1697 -                 ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
1698 -                 ang_washout(ind2) = ang_washout(ind2)-180;
1699 -                 ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
1700 -                 ang_washout(ind3) = 180;
1701 -                 [list,ind_washout] = sort(ang_washout);
1702 -                 ind_sort_targ_washout = ind_washout;
1703 -                 targets_sort_washout = targets_washout(ind_sort_targ_washout,:);
1704 -
1705 -                 angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
1706 -                 app.angle_washout = angle_rad_washout * 360 / (2*pi);
1707 -                 ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
1708 -                 app.angle_washout(ind1) = app.angle_washout(ind1)+180;
1709 -                 ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
1710 -                 app.angle_washout(ind2) = app.angle_washout(ind2)-180;
1711 -                 ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
1712 -                 app.angle_washout(ind3) = 180;
1713 -
1714 -                 sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
1715 -                 sr_cell_washout = sort_rates_washout(:,cell_washout);
1716 -                 app.rates_washout(:,i) = sr_cell_washout';
1717 -
1718 -                 early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
1719 -             end
1720 -         end
1721 -     end

```

```

1719
1720         end
1721         app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
1722     catch err
1723         %errorlog(err.message)
1724     end
1725 end
1726 try
1727     mean_all_rates_washout = mean(app.all_rates_washout');
1728     [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
1729     angle_max_washout = app.angle_washout(index_max_value_washout);
1730     sentence_washout = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_washout, ..
1731         max_value_washout);
1732     app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
1733     ax_washout = uiaxes(panel2, 'Position', [10, 140*(num_days-day_i)+10, 688, 130]);
1734     hold(ax_washout, 'on')
1735     plot(ax_washout, app.angle_washout, app.all_rates_washout, 'b.', 'MarkerSize', 12)
1736     plot(ax_washout, app.angle_washout, mean_all_rates_washout, 'LineWidth', 4)
1737     plot(ax_washout, angle_max_washout, max_value_washout, 'r.', 'MarkerSize', 12)
1738     xlabel(ax_washout, 'Angle', 'FontSize', 12)
1739     ylabel(ax_washout, 'Spiking rate', 'FontSize', 12)
1740     title(ax_washout, strcat('Day: ', days(day_i)))
1741     hold(ax_washout, 'off')
1742 catch err
1743     %errorlog(err.message)
1744 end
1745 end
1746 else
1747     panel1.Title = 'Perturbation';
1748     panel1.FontSize = 16;
1749     panel1.FontWeight = 'Bold';
1750     for file = 2:path_length_perturbation
1751     try
1752         app.rates_perturb = [];
1753         ang = [];
1754         i=1;
1755         if isempty(path_perturbation)
1756         else
1757             app.id_perturbation = num2str(path_perturbation(file));
1758             if numel(app.id_perturbation) == 4
1759                 start_fname = 'Arthur.BC.0';
1760             else
1761                 start_fname = 'Arthur.BC.';
1762             end
1763             app.path_file_perturb = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_perturbation(file)), end_fname);
1764             e_perturb = load(app.path_file_perturb);
1765
1766 [info_perturb, target_perturb, app.cellnames_perturb, celldata_perturb] = get_seq_rate_info_short_win(app, app.path_file_perturb, ...
1767     ['half_rt', 'rate', 0.3]);
1768
1769 app.cell_ind_perturb = [];
1770 for cname = 1:length(app.cellnames_perturb)
1771     cell_val_perturb = app.cellnames_perturb{cname};
1772     if cell_val_perturb == target_cell
1773         app.cell_ind_perturb = cname;
1774     end
1775 end
1776 if isempty(app.cell_ind_perturb)
1777     app.cell_ind_perturb = 1;
1778 end
1779 for cell_perturb = app.cell_ind_perturb
1780     targets_perturb = e_perturb.trials.TargetPos(:,1:2);
1781     angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
1782     ang_perturb = angles_rad_perturb * 360 / (2*pi);
1783     ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
1784     ang_perturb(ind1) = ang_perturb(ind1)+180;
1785     ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
1786     ang_perturb(ind2) = ang_perturb(ind2)-180;
1787     ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
1788     ang_perturb(ind3) = 180;
1789     [list, ind_perturb] = sort(ang_perturb);
1790     ind_sort_targ_perturb = ind_perturb;
1791     targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);
1792
1793     angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
1794     app.angle_perturb = angle_rad_perturb * 360 / (2*pi);
1795     ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
1796     app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
1797     ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
1798     app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
1799     ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
1800     app.angle_perturb(ind3) = 180;
1801
1802     sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
1803     sr_cell_perturb = sort_rates_perturb(:, cell_perturb);
1804     app.rates_perturb(:,i) = sr_cell_perturb';
1805
1806     early_perturb_fit = compute_cosine_fit(targets_perturb, app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
1807 end
1808 end

```

```

1808 -         app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
1809 -     catch err
1810 -         %errorlog(err.message)
1811 -     end
1812 - end
1813 - try
1814 -
1815 -     mean_all_rates_perturb = mean(app.all_rates_perturb');
1816 -     [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
1817 -     angle_max_perturb = app.angle_perturb(index_max_value_perturb);
1818 -     sentence_perturb = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_perturb, ...
1819 -         max_value_perturb);
1820 -     app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
1821 -     ax_perturb = uiaxes(panel1,'Position',[10, 140*(num_days-day_i)+10,688,130]);
1822 -     hold(ax_perturb, 'on')
1823 -     plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
1824 -     plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
1825 -     plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
1826 -     xlabel(ax_perturb,'Angle','FontSize',12)
1827 -     ylabel(ax_perturb, 'Spiking rate','FontSize',12)
1828 -     title(ax_perturb, strcat('Day: ', days(day_i)))
1829 -     hold(ax_perturb,"off")
1830 - catch err
1831 -     %errorlog(err.message)
1832 - end
1833 - panel2.Title = 'Washout';
1834 - panel2.FontSize = 16;
1835 - panel2.FontHeight = 'Bold';
1836 - for file = 2:path_length_washout
1837 -     try
1838 -         app.rates_washout = [];
1839 -         ang = [];
1840 -         i=1;
1841 -         if isempty(path_washout)
1842 -             else
1843 -                 app.id_washout = num2str(path_washout(file));
1844 -                 if numel(app.id_washout) == 4
1845 -                     start_fname = 'Arthur.BC.0';
1846 -                 else
1847 -                     start_fname = 'Arthur.BC.';
1848 -                 end
1849 -
1850 -                 app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
1851 -                 e_washout = load(app.path_file_washout);
1852 -                 [info_washout,targetset_washout,app.cellnames_washout,celldata_washout] = get_seq_rate_info_short_win(app, app.path_file_washout, ...
1853 -                     'half_rt','rate',0.3);
1854 -
1855 -                 app.cell_ind_washout = [];
1856 -                 for cname = 1:length(app.cellnames_washout) %cellnames_valid
1857 -                     cell_val_washout = app.cellnames_washout(cname); % coge el numero de unidad --> 18_1
1858 -                     if cell_val_washout == target_cell
1859 -                         app.cell_ind_washout = cname;
1860 -                     end
1861 -                 end
1862 -                 if isempty(app.cell_ind_washout)
1863 -                     app.cell_ind_washout = 1;
1864 -                 end
1865 -                 for cell_washout = app.cell_ind_washout
1866 -                     targets_washout = e_washout.trials.TargetPos(:,1:2);
1867 -                     angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
1868 -                     ang_washout = angles_rad_washout * 360 / (2*pi);
1869 -                     ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
1870 -                     ang_washout(ind1) = ang_washout(ind1)+180;
1871 -                     ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
1872 -                     ang_washout(ind2) = ang_washout(ind2)-180;
1873 -                     ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
1874 -                     ang_washout(ind3) = 180;
1875 -                     [list,ind_washout] = sort(ang_washout);
1876 -                     ind_sort_targ_washout = ind_washout;
1877 -                     targets_sort_washout = targets_washout(ind_sort_targ_washout,:);
1878 -
1879 -                     angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
1880 -                     app.angle_washout = angle_rad_washout * 360 / (2*pi);
1881 -                     ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
1882 -                     app.angle_washout(ind1) = app.angle_washout(ind1)+180;
1883 -                     ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
1884 -                     app.angle_washout(ind2) = app.angle_washout(ind2)-180;
1885 -                     ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
1886 -                     app.angle_washout(ind3) = 180;
1887 -
1888 -                     sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
1889 -                     sr_cell_washout = sort_rates_washout(:,cell_washout);
1890 -                     app.rates_washout(:,i) = sr_cell_washout';
1891 -
1892 -                     early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
1893 -                 end
1894 -             end

```

```

1894 -         app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
1895 -     catch err
1896 -         %errorrdlg(err.message)
1897 -     end
1898 - end
1899 - try
1900 -     mean_all_rates_washout = mean(app.all_rates_washout');
1901 -     [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
1902 -     angle_max_washout = app.angle_washout(index_max_value_washout);
1903 -     sentence_washout = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_washout, ...
1904 -         [max_value_washout]);
1905 -     app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
1906 -     ax_washout = uiaxes(panel2, 'Position', [10, 140*(num_days-day_i)+10, 688, 130]);
1907 -     hold(ax_washout, 'on')
1908 -     plot(ax_washout, app.angle_washout, app.all_rates_washout, 'b.', 'MarkerSize', 12)
1909 -     plot(ax_washout, app.angle_washout, mean_all_rates_washout, 'LineWidth', 4)
1910 -     plot(ax_washout, angle_max_washout, max_value_washout, 'r.', 'MarkerSize', 12)
1911 -     xlabel(ax_washout, 'Angle', 'FontSize', 12)
1912 -     ylabel(ax_washout, 'Spiking rate', 'FontSize', 12)
1913 -     title(ax_washout, strcat('Day: ', days(day_i)))
1914 -     hold(ax_washout, 'off')
1915 -     catch err
1916 -         %errorrdlg(err.message)
1917 -     end
1918 - end
1919 - end
1920 - else
1921 -     panel1 = uipanel(app.TCWindow, 'Position', [21, 40, 484, 560], 'Scrollable', 'on' );
1922 -     panel2 = uipanel(app.TCWindow, 'Position', [526, 40, 484, 560], 'Scrollable', 'on' );
1923 -     panel3 = uipanel(app.TCWindow, 'Position', [1031, 40, 484, 560], 'Scrollable', 'on' );
1924 -     for day_i = day_s:day_e
1925 -         app.all_rates_control = [];
1926 -         app.all_rates_perturb = [];
1927 -         app.all_rates_washout = [];
1928 -         path_file_inde = strcat(string(path), '\', string(days(day_i)), '\file_inde.mat');
1929 -         d = load(path_file_inde);
1930 -         Const = fieldnames(d);
1931 -         path_control = d.(Const{1});
1932 -         path_length_control = length(path_control);
1933 -         path_perturbation = d.(Const{2});
1934 -         path_length_perturbation = length(path_perturbation);
1935 -         path_washout = d.(Const{3});
1936 -         path_length_washout = length(path_washout);
1937 -
1938 -         panel1.Title = 'Control';
1939 -         panel1.FontSize = 16;
1940 -         panel1.FontWeight = 'Bold';
1941 -         for file = 2:path_length_control
1942 -             try
1943 -                 app.rates_control = [];
1944 -                 ang = [];
1945 -                 i=1;
1946 -                 if isempty(path_control)
1947 -                     else
1948 -                         app.id_control = num2str(path_control(file));
1949 -                         if numel(app.id_control) == 4
1950 -                             start_fname = 'Arthur.BC.0';
1951 -                         else
1952 -                             start_fname = 'Arthur.BC.';
1953 -                         end
1954 -                         app.path_file_control = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_control(file)), end_fname);
1955 -                         e_control = load(app.path_file_control);
1956 -
1957 -                         [info_control, target_control, app.cellnames_control, celldata_control] = get_seq_rate_info_short_win(app, app.path_file_control, ...
1958 -                             'half_rt', 'rate', 0.3);
1959 -
1960 -                         app.cell_ind_control = [];
1961 -                         for cname = 1:length(app.cellnames_control)
1962 -                             cell_val_control = app.cellnames_control(cname);
1963 -                             if cell_val_control == target_cell
1964 -                                 app.cell_ind_control = cname;
1965 -                             end
1966 -                         end
1967 -                         if isempty(app.cell_ind_control)
1968 -                             app.cell_ind_control = 1;
1969 -                         end
1970 -                         for cell_control = app.cell_ind_control
1971 -                             targets_control = e_control.trials.TargetPos(:, 1:2);
1972 -                             angles_rad_control = atan(targets_control(:, 2)./targets_control(:, 1));
1973 -                             ang_control = angles_rad_control * 360 / (2*pi);
1974 -                             ind1 = find(targets_control(:, 1)<0 & targets_control(:, 2)>0);
1975 -                             ang_control(ind1) = ang_control(ind1)+180;
1976 -                             ind2 = find(targets_control(:, 1)>0 & targets_control(:, 2)<0);
1977 -                             ang_control(ind2) = ang_control(ind2)-180;
1978 -                             ind3 = find(targets_control(:, 1)<0 & targets_control(:, 2)==0);
1979 -                             ang_control(ind3) = 180;
1980 -                             [list, ind_control] = sort(ang_control);
1981 -                             ind_sort_targ_control = ind_control;

```

```

1981 - targets_sort_control = targets_control(ind_sort_targ_control,:);
1982 -
1983 - angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
1984 - app.angle_control = angle_rad_control * 360 / (2*pi);
1985 - ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
1986 - app.angle_control(ind1) = app.angle_control(ind1)+180;
1987 - ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
1988 - app.angle_control(ind2) = app.angle_control(ind2)-180;
1989 - ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
1990 - app.angle_control(ind3) = 180;
1991 -
1992 - sort_rates_control = info_control.rates(ind_sort_targ_control,:);
1993 - sr_cell_control = sort_rates_control(:,cell_control);
1994 - app.rates_control(:,i) = sr_cell_control';
1995 -
1996 - early_control_fit = compute_cosine_fit(targets_control,app.rates_control); % Obtain the parameters needed to fit cosine tuning curves
1997 - end
1998 - end
1999 - app.all_rates_control = [app.all_rates_control sr_cell_control];
2000 - catch err
2001 - %errorlog(err.message)
2002 - end
2003 - end
2004 - try
2005 - mean_all_rates_control = mean(app.all_rates_control');
2006 - [max_value_control, index_max_value_control] = max(mean_all_rates_control');
2007 - angle_max_control = app.angle_control(index_max_value_control);
2008 - sentence_control = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_control, ...
2009 - max_value_control);
2010 - app.all_sentences_control = [app.all_sentences_control, sentence_control];
2011 - ax_control = uiaxes(panel1,'Position',[10, 140*(num_days-day_i)+10,464,130]);
2012 - hold(ax_control, 'on')
2013 - plot(ax_control, app.angle_control, app.all_rates_control, 'b.', 'MarkerSize', 12)
2014 - plot(ax_control, app.angle_control, mean_all_rates_control, 'LineWidth', 4)
2015 - plot(ax_control, angle_max_control, max_value_control, 'r.', 'MarkerSize', 12)
2016 - xlabel(ax_control, 'Angle', 'FontSize', 12)
2017 - ylabel(ax_control, 'Spiking rate', 'FontSize', 12)
2018 - title(ax_control, strcat('Day: ', days(day_i)))
2019 - hold(ax_control, "off")
2020 - catch err
2021 - %errorlog(err.message)ME
2022 - end

2023 - panel2.Title = 'Perturbation';
2024 - panel2.FontSize = 16;
2025 - panel2.FontWeight = 'Bold';
2026 - for file = 2:path_length_perturbation
2027 - try
2028 - app.rates_perturb = [];
2029 - ang = [];
2030 - i=1;
2031 - if isempty(path_perturbation)
2032 - else
2033 - app.id_perturbation = num2str(path_perturbation(file));
2034 - if numel(app.id_perturbation) == 4
2035 - start_fname = 'Arthur.BC.0';
2036 - else
2037 - start_fname = 'Arthur.BC.';
2038 - end
2039 - app.path_file_perturb = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_perturbation(file)), end_fname);
2040 - e_perturb = load(app.path_file_perturb);
2041 -
2042 - [info_perturb, target_perturb, app.cellnames_perturb, celldata_perturb] = get_seq_rate_info_short_win(app, app.path_file_perturb, ...
2043 - |half_rt', 'rate', 0.3);
2044 -
2045 - app.cell_ind_perturb = [];
2046 - for cname = 1:length(app.cellnames_perturb)
2047 - cell_val_perturb = app.cellnames_perturb(cname);
2048 - if cell_val_perturb == target_cell
2049 - app.cell_ind_perturb = cname;
2050 - end
2051 - end
2052 - if isempty(app.cell_ind_perturb)
2053 - app.cell_ind_perturb = 1;
2054 - end
2055 - for cell_perturb = app.cell_ind_perturb
2056 - targets_perturb = e_perturb.trials.TargetPos(:,1:2);
2057 - angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
2058 - ang_perturb = angles_rad_perturb * 360 / (2*pi);
2059 - ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
2060 - ang_perturb(ind1) = ang_perturb(ind1)+180;
2061 - ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
2062 - ang_perturb(ind2) = ang_perturb(ind2)-180;
2063 - ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
2064 - ang_perturb(ind3) = 180;
2065 - [list, ind_perturb] = sort(ang_perturb);
2066 - ind_sort_targ_perturb = ind_perturb;
2067 - targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);

```

```

2068
2069 -         angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
2070 -         app.angle_perturb = angle_rad_perturb * 360 / (2*pi);
2071 -         ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
2072 -         app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
2073 -         ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
2074 -         app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
2075 -         ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
2076 -         app.angle_perturb(ind3) = 180;
2077
2078 -         sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
2079 -         sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
2080 -         app.rates_perturb(:,i) = sr_cell_perturb';
2081
2082 -         early_perturb_fit = compute_cosine_fit(targets_perturb,app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
2083 -     end
2084 - end
2085 - app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
2086 - catch err
2087 -     %errordlg(err.message)
2088 - end
2089 - end
2090 - try
2091 -     mean_all_rates_perturb = mean(app.all_rates_perturb');
2092 -     [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
2093 -     angle_max_perturb = app.angle_perturb(index_max_value_perturb);
2094 -     sentence_perturb = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_perturb, ...
2095 -         max_value_perturb);
2096 -     app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
2097 -     ax_perturb = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,464,130]);
2098 -     hold(ax_perturb, 'on')
2099 -     plot(ax_perturb, app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
2100 -     plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
2101 -     plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
2102 -     xlabel(ax_perturb,'Angle','FontSize',12)
2103 -     ylabel(ax_perturb, 'Spiking rate','FontSize',12)
2104 -     title(ax_perturb, strcat('Day: ', days(day_i)))
2105 -     hold(ax_perturb,'off')
2106 - catch err
2107 -     %errordlg(err.message)
2108 - end
2109 -
2110 - panel3.Title = 'Washout';
2111 - panel3.FontSize = 16;
2112 - panel3.FontWeight = 'Bold';
2113 - for file = 2:path_length_washout
2114 -     try
2115 -         app.rates_washout = [];
2116 -         ang = [];
2117 -         i=1;
2118 -         if isempty(path_washout)
2119 -             else
2120 -                 app.id_washout = num2str(path_washout(file));
2121 -                 if numel(app.id_washout) == 4
2122 -                     start_fname = 'Arthur.BC.0';
2123 -                 else
2124 -                     start_fname = 'Arthur.BC.';
2125 -                 end
2126 -                 app.path_file_washout = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(path_washout(file)), end_fname);
2127 -                 e_washout = load(app.path_file_washout);
2128 -                 [info_washout, targetset_washout, app.cellnames_washout, celldata_washout] = get_seq_rate_info_short_win(app, app.path_file_washout, ...
2129 -                     'half_rt','rate',0.3);
2130 -
2131 -                 app.cell_ind_washout = [];
2132 -                 for cname = 1:length(app.cellnames_washout)
2133 -                     cell_val_washout = app.cellnames_washout{cname};
2134 -                     if cell_val_washout == target_cell
2135 -                         app.cell_ind_washout = cname;
2136 -                     end
2137 -                 end
2138 -                 if isempty(app.cell_ind_washout)
2139 -                     app.cell_ind_washout = 1;
2140 -                 end
2141 -                 cell_washout = app.cell_ind_washout
2142 -                 targets_washout = e_washout.trials.TargetPos(:,1:2);
2143 -                 angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
2144 -                 ang_washout = angles_rad_washout * 360 / (2*pi);
2145 -                 ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
2146 -                 ang_washout(ind1) = ang_washout(ind1)+180;
2147 -                 ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
2148 -                 ang_washout(ind2) = ang_washout(ind2)-180;
2149 -                 ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
2150 -                 ang_washout(ind3) = 180;
2151 -                 [list, ind_washout] = sort(ang_washout);
2152 -                 ind_sort_targ_washout = ind_washout;
2153 -                 targets_sort_washout = targets_washout(ind_sort_targ_washout,:);

```

```

2154
2155     angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
2156     app.angle_washout = angle_rad_washout * 360 / (2*pi);
2157     ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
2158     app.angle_washout(ind1) = app.angle_washout(ind1)+180;
2159     ind2 = find(targets_sort_washout(:,1)>0 & targets_sort_washout(:,2)<0);
2160     app.angle_washout(ind2) = app.angle_washout(ind2)-180;
2161     ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
2162     app.angle_washout(ind3) = 180;
2163
2164     sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
2165     sr_cell_washout = sort_rates_washout(:,cell_washout);
2166     app.rates_washout(:,i) = sr_cell_washout';
2167
2168     early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
2169
2170     end
2171     app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
2172     catch err
2173         %erroridg(err.message)
2174     end
2175 end
2176 try
2177     mean_all_rates_washout = mean(app.all_rates_washout');
2178     [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
2179     angle_max_washout = app.angle_washout(index_max_value_washout);
2180     sentence_washout = sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_washout, ...
2181         |max_value_washout);
2182     app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
2183     ax_washout = uiaxes(panel3,'Position',[10, 140*(num_days-day_i)+10,464,130]);
2184     hold(ax_washout, 'on')
2185     plot(ax_washout,app.angle_washout,app.all_rates_washout,'b','MarkerSize',12)
2186     plot(ax_washout,app.angle_washout,mean_all_rates_washout,'LineWidth',4)
2187     plot(ax_washout, angle_max_washout,max_value_washout,'r.', 'MarkerSize', 12)
2188     xlabel(ax_washout,'Angle','FontSize',12)
2189     ylabel(ax_washout,'Spiking rate','FontSize',12)
2190     title(ax_washout, strcat('Day: ', days(day_i)))
2191     hold(ax_washout,'off')
2192     catch err
2193         %erroridg(err.message)
2194     end
2195 end
2196
2197 if ~isempty(app.all_sentences_control)
2198     app.callerApp_sent_control = TFG_Sentences_Control(app.all_sentences_control);
2199 end
2200 if ~isempty(app.all_sentences_perturbation)
2201     app.callerApp_sent_perturbation = TFG_Sentences_Perturbation(app.all_sentences_perturbation);
2202 end
2203 if ~isempty(app.all_sentences_washout)
2204     app.callerApp_sent_washout = TFG_Sentences_Washout(app.all_sentences_washout);
2205 end
2206
2207 end
2208
2209 % Value changed function: SaveScreenshotButton
2210 function SaveScreenshotButtonValueChanged(app, event)
2211
2212     % Save a screenshot of the window
2213     filter = {'*.jpg'; '*.png'; '*.tif'; '*.pdf'}; % File type options
2214     [filename,filepath] = uiputfile(filter); % Open dialog box for saving files
2215     if ischar(filename) % If the name has the correct format, save the file
2216         exportapp(app.TCWindow,[filepath filename]);
2217     end
2218     app.TCWindow.Visible = 'off'; % These two lines of code work-around an issue whether the figure is sent to the background.
2219     app.TCWindow.Visible = 'on';
2220
2221 end
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500

```


Component initialization

```

2225 methods (Access = private)
2226
2227 % Create UIFigure and components
2228 function createComponents(app)
2229
2230 % Create TCWindow and hide until all components are created
2231 app.TCWindow = uifigure('Visible', 'off');
2232 app.TCWindow.Position = [0 40 1536 800];
2233 app.TCWindow.Name = 'MATLAB App';
2234
2235 % Create TargetCellPanel
2236 app.TargetCellPanel = uipanel(app.TCWindow);
2237 app.TargetCellPanel.TitlePosition = 'centertop';
2238 app.TargetCellPanel.Title = 'Target Cell';
2239 app.TargetCellPanel.BackgroundColor = [1 1 1];
2240 app.TargetCellPanel.Position = [1031 612 484 178];
2241
2242 % Create SaveScreenshotButton
2243 app.SaveScreenshotButton = uibutton(app.TargetCellPanel, 'state');
2244 app.SaveScreenshotButton.ValueChangedFcn = createCallbackFcn(app, @SaveScreenshotButtonValueChanged, true);
2245 app.SaveScreenshotButton.Text = 'Save Screenshot';
2246 app.SaveScreenshotButton.FontWeight = 'bold';
2247 app.SaveScreenshotButton.Position = [310 18 112 22];
2248
2249 % Create SelectTCButton
2250 app.SelectTCButton = uibutton(app.TargetCellPanel, 'push');
2251 app.SelectTCButton.ButtonPushedFcn = createCallbackFcn(app, @SelectTCButtonPushed, true);
2252 app.SelectTCButton.Position = [58 8 100 22];
2253 app.SelectTCButton.Text = 'Select TC';
2254
2255 % Create ShowTargetCellOptionsButton
2256 app.ShowTargetCellOptionsButton = uibutton(app.TargetCellPanel, 'push');
2257 app.ShowTargetCellOptionsButton.ButtonPushedFcn = createCallbackFcn(app, @ShowTargetCellOptionsButtonPushed, true);
2258 app.ShowTargetCellOptionsButton.Position = [36 130 151 22];
2259 app.ShowTargetCellOptionsButton.Text = 'Show Target Cell Options';
2260
2261 % Create SelectedTCTextAreaLabel
2262 app.SelectedTCTextAreaLabel = uilabel(app.TargetCellPanel);
2263 app.SelectedTCTextAreaLabel.HorizontalAlignment = 'right';
2264 app.SelectedTCTextAreaLabel.Position = [274 83 75 22];
2265 app.SelectedTCTextAreaLabel.Text = 'Selected TC: ';
2266 app.SelectedTCTextArea = uitextarea(app.TargetCellPanel);
2267 app.SelectedTCTextArea.Position = [353 80 105 25];
2268
2269 % Create NumTCAvailableTextAreaLabel
2270 app.NumTCAvailableTextAreaLabel = uilabel(app.TargetCellPanel);
2271 app.NumTCAvailableTextAreaLabel.HorizontalAlignment = 'right';
2272 app.NumTCAvailableTextAreaLabel.Position = [274 114 105 22];
2273 app.NumTCAvailableTextAreaLabel.Text = 'Num. TC Available';
2274
2275 % Create NumTCAvailableTextArea
2276 app.NumTCAvailableTextArea = uitextarea(app.TargetCellPanel);
2277 app.NumTCAvailableTextArea.Editable = 'off';
2278 app.NumTCAvailableTextArea.Position = [387 113 33 24];
2279
2280 % Create ChooseTargetCellListBoxLabel
2281 app.ChooseTargetCellListBoxLabel = uilabel(app.TargetCellPanel);
2282 app.ChooseTargetCellListBoxLabel.Position = [47 105 111 22];
2283 app.ChooseTargetCellListBoxLabel.Text = 'Choose Target Cell: ';
2284
2285 % Create ChooseTargetCellListBox
2286 app.ChooseTargetCellListBox = uilistbox(app.TargetCellPanel);
2287 app.ChooseTargetCellListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseTargetCellListBoxValueChanged, true);
2288 app.ChooseTargetCellListBox.Position = [47 39 129 66];
2289
2290 % Create PhasesPanel
2291 app.PhasesPanel = uipanel(app.TCWindow);
2292 app.PhasesPanel.TitlePosition = 'centertop';
2293 app.PhasesPanel.Title = 'Phases';
2294 app.PhasesPanel.BackgroundColor = [1 1 1];
2295 app.PhasesPanel.Position = [526 612 484 178];
2296
2297 % Create DeleteAllPhasesButton
2298 app.DeleteAllPhasesButton = uibutton(app.PhasesPanel, 'push');
2299 app.DeleteAllPhasesButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllPhasesButtonPushed, true);
2300 app.DeleteAllPhasesButton.FontSize = 8;
2301 app.DeleteAllPhasesButton.Position = [348 33 76 22];
2302 app.DeleteAllPhasesButton.Text = 'Delete All Phases';
2303
2304
2305

```

```

2308 -     app.DeletePhaseButton.ButtonPushedFcn = createCallbackFcn(app, @DeletePhaseButtonPushed, true);
2309 -     app.DeletePhaseButton.FontSize = 8;
2310 -     app.DeletePhaseButton.Position = [277 33 62 22];
2311 -     app.DeletePhaseButton.Text = 'Delete Phase';
2312
2313 -     % Create SelectAllPhasesButton
2314 -     app.SelectAllPhasesButton = uibutton(app.PhasesPanel, 'push');
2315 -     app.SelectAllPhasesButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllPhasesButtonPushed, true);
2316 -     app.SelectAllPhasesButton.FontSize = 8;
2317 -     app.SelectAllPhasesButton.Position = [100 8 90 22];
2318 -     app.SelectAllPhasesButton.Text = 'Select All Phases';
2319
2320 -     % Create SelectPhaseButton
2321 -     app.SelectPhaseButton = uibutton(app.PhasesPanel, 'push');
2322 -     app.SelectPhaseButton.ButtonPushedFcn = createCallbackFcn(app, @SelectPhaseButtonPushed, true);
2323 -     app.SelectPhaseButton.BackgroundColor = [1 1 1];
2324 -     app.SelectPhaseButton.Position = [95 35 100 22];
2325 -     app.SelectPhaseButton.Text = 'Select Phase';
2326
2327 -     % Create SelectedPhasesListBoxLabel
2328 -     app.SelectedPhasesListBoxLabel = uilabel(app.PhasesPanel);
2329 -     app.SelectedPhasesListBoxLabel.Position = [289 136 99 22];
2330 -     app.SelectedPhasesListBoxLabel.Text = 'Selected Phases: ';
2331
2332 -     % Create SelectedPhasesListBox
2333 -     app.SelectedPhasesListBox = uilistbox(app.PhasesPanel);
2334 -     app.SelectedPhasesListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedPhasesListBoxValueChanged, true);
2335 -     app.SelectedPhasesListBox.Position = [289 62 100 74];
2336
2337 -     % Create ChoosePhasesListBoxLabel
2338 -     app.ChoosePhasesListBoxLabel = uilabel(app.PhasesPanel);
2339 -     app.ChoosePhasesListBoxLabel.Position = [95 136 94 22];
2340 -     app.ChoosePhasesListBoxLabel.Text = 'Choose Phases: ';
2341
2342 -     % Create ChoosePhasesListBox
2343 -     app.ChoosePhasesListBox = uilistbox(app.PhasesPanel);
2344 -     app.ChoosePhasesListBox.ValueChangedFcn = createCallbackFcn(app, @ChoosePhasesListBoxValueChanged, true);
2345 -     app.ChoosePhasesListBox.Position = [95 63 100 74];
2346 -     app.DaysPanel = uipanel(app.TCWindow);
2347 -     app.DaysPanel.TitlePosition = 'centertop';
2348 -     app.DaysPanel.Title = 'Days';
2349 -     app.DaysPanel.BackgroundColor = [1 1 1];
2350 -     app.DaysPanel.Position = [21 612 484 178];
2351
2352 -     % Create DeleteAllDaysButton
2353 -     app.DeleteAllDaysButton = uibutton(app.DaysPanel, 'push');
2354 -     app.DeleteAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllDaysButtonPushed, true);
2355 -     app.DeleteAllDaysButton.FontSize = 8;
2356 -     app.DeleteAllDaysButton.Position = [352 39 71 20];
2357 -     app.DeleteAllDaysButton.Text = 'Delete All Days';
2358
2359 -     % Create DeleteDayButton
2360 -     app.DeleteDayButton = uibutton(app.DaysPanel, 'push');
2361 -     app.DeleteDayButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteDayButtonPushed, true);
2362 -     app.DeleteDayButton.FontSize = 8;
2363 -     app.DeleteDayButton.Position = [287 39 51 20];
2364 -     app.DeleteDayButton.Text = 'Delete Day';
2365
2366 -     % Create SelectAllDaysButton
2367 -     app.SelectAllDaysButton = uibutton(app.DaysPanel, 'push');
2368 -     app.SelectAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllDaysButtonPushed, true);
2369 -     app.SelectAllDaysButton.FontSize = 8;
2370 -     app.SelectAllDaysButton.Position = [98 8 67 22];
2371 -     app.SelectAllDaysButton.Text = 'Select All Days';
2372
2373 -     % Create SelectDayButton
2374 -     app.SelectDayButton = uibutton(app.DaysPanel, 'push');
2375 -     app.SelectDayButton.ButtonPushedFcn = createCallbackFcn(app, @SelectDayButtonPushed, true);
2376 -     app.SelectDayButton.BackgroundColor = [1 1 1];
2377 -     app.SelectDayButton.Position = [82 36 100 22];
2378 -     app.SelectDayButton.Text = 'Select Day';
2379
2380 -     % Create SelectedDaysListBoxLabel
2381 -     app.SelectedDaysListBoxLabel = uilabel(app.DaysPanel);
2382 -     app.SelectedDaysListBoxLabel.Position = [287 136 86 22];
2383 -     app.SelectedDaysListBoxLabel.Text = 'Selected Days: ';
2384
2385 -     app.SelectedDaysListBox.Multiselect = 'on';
2386 -     app.SelectedDaysListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedDaysListBoxValueChanged, true);
2387 -     app.SelectedDaysListBox.Position = [287 63 136 72];
2388 -     app.SelectedDaysListBox.Value = {'Item 1'};
2389
2390 -     % Create ChooseDaysListBoxLabel
2391 -     app.ChooseDaysListBoxLabel = uilabel(app.DaysPanel);
2392 -     app.ChooseDaysListBoxLabel.Position = [61 136 81 22];
2393 -     app.ChooseDaysListBoxLabel.Text = 'Choose Days: ';
2394
2395 -     % Create ChooseDaysListBox
2396 -     app.ChooseDaysListBox = uilistbox(app.DaysPanel);
2397 -     app.ChooseDaysListBox.Multiselect = 'on';
2398 -     app.ChooseDaysListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseDaysListBoxValueChanged, true);
2399 -     app.ChooseDaysListBox.Position = [61 63 136 72];
2400 -     app.ChooseDaysListBox.Value = {'Item 1'};
2401
2402 -     % Show the figure after all components are created
2403 -     app.TCWindow.Visible = 'on';
2404
2405 - end
2406
2407 - end
2408
2409

```

```
2410
2411 % App creation and deletion
2412 methods (Access = public)
2413
2414 % Construct app
2415 function app = TC_FINAL(varargin)
2416
2417     % Create UIFigure and components
2418     createComponents(app)
2419
2420     % Register the app with App Designer
2421     registerApp(app, app.TCWindow)
2422
2423     % Execute the startup function
2424     runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
2425
2426     if nargin == 0
2427         clear app
2428     end
2429 end
2430
2431 % Code that executes before app deletion
2432 function delete(app)
2433
2434     % Delete UIFigure when app is deleted
2435     delete(app.TCWindow)
2436 end
2437 end
2438
```

ANNEX III. Code used for the development of the Trajectories window

Properties

```

3 % Properties that correspond to app components
4 properties (Access = public)
5     TrajectoriesWindow          matlab.ui.Figure
6     TargetCellPanel            matlab.ui.container.Panel
7     SelectedTCTextAreaLabel    matlab.ui.control.Label
8     SelectedTCTextArea         matlab.ui.control.TextArea
9     NumTCAvailableTextAreaLabel matlab.ui.control.Label
10    NumTCAvailableTextArea     matlab.ui.control.TextArea
11    SelectTCButton              matlab.ui.control.Button
12    ChooseTargetCellListBoxLabel matlab.ui.control.Label
13    ChooseTargetCellListBox     matlab.ui.control.ListBox
14    ShowTargetCellOptionsButton matlab.ui.control.Button
15    SaveScreenshotButton        matlab.ui.control.StateButton
16    PhasePanel                  matlab.ui.container.Panel
17    SelectPhaseButton           matlab.ui.control.Button
18    SelectedPhaseListBoxLabel    matlab.ui.control.Label
19    SelectedPhaseListBox        matlab.ui.control.ListBox
20    ChoosePhaseListBoxLabel     matlab.ui.control.Label
21    ChoosePhaseListBox          matlab.ui.control.ListBox
22    DaysPanel                   matlab.ui.container.Panel
23    DeleteAllDaysButton          matlab.ui.control.Button
24    DeleteDayButton              matlab.ui.control.Button
25    SelectAllDaysButton         matlab.ui.control.Button
26    SelectDayButton             matlab.ui.control.Button
27    SelectedDaysListBoxLabel     matlab.ui.control.Label
28    SelectedDaysListBox         matlab.ui.control.ListBox
29    ChoosedaysListBoxLabel      matlab.ui.control.Label
30    ChoosedaysListBox           matlab.ui.control.ListBox
31 end
32 %% Variables used when selecting the options
33 selectedPath = '' % Path selected by user in ButtonPushed function
34 day           % Day selected in ChooseDaysListBox
35 total_days   % All days selected
36 value_day    % Selected day to delete
37 phase_selected % Selected phase
38 target_cell  % Selected target cell
39 phases       % Available phases
40
41 %% Variables used when generating the target cells list
42 id           % Identification number of a file
43 C = {}       % All target cells
44 target_cells_list = [] % Target cell list shown in ChooseTargetCellListBox (common TC)
45
46 %% Variables used when obtaining and plotting the trajectories
47 path_phase   % File directory
48 targs        % Target positions
49 traj         % Positions of the trajectories
50
51 end

```

Startup function

```

56 % Callbacks that handle component events
57 methods (Access = private)
58
59 % Code that executes after component creation
60 function startupFcn(app, path)
61
62 % Set window name and only show in the day panel
63 % ChooseDayListBox and the buttons for selecting the days
64 app.TrajectoriesWindow.Name = 'Trajectories';
65 app.SelectedDaysListBox.Visible = 'off';
66 app.SelectedDaysListBoxLabel.Visible = 'off';
67 app.DeleteDayButton.Visible = 'off';
68 app.DeleteAllDaysButton.Visible = 'off';
69 app.TargetCellPanel.Visible = 'off';
70 app.ChooseTargetCellListBox.Visible = 'off';
71 app.ChooseTargetCellListBoxLabel.Visible = 'off';
72 app.SelectTCButton.Visible = 'off';
73 app.SelectedTCTextArea.Visible = 'off';
74 app.SelectedTCTextAreaLabel.Visible = 'off';
75 app.NumTCAvailableTextArea.Visible = 'off';
76 app.NumTCAvailableTextAreaLabel.Visible = 'off';
77 app.PhasePanel.Visible = 'off';
78 app.SelectedPhaseListBox.Visible = 'off';
79 app.SelectedPhaseListBoxLabel.Visible = 'off';

```

```

81 -
82 -     app.selectedPath = path;           % Save the path selected in the main window in app.selectedPath
83 -     a = dir(app.selectedPath);        % Obtain the contents of the selected path.
84 -     b = {a(:).name};                 % Get the name of the folders of the contents and stores them appropriately in a cell array
85 -     b(ismember(b,{'.','..'})==0) = []; % Remove unnecessary '.' and '..' results from the display.
86 -
87 - % Create a vector days with only the days whose files can be open
88 - % without error
89 -     days = {};
90 -     for element = 1:length(b)
91 -         val = b(element);
92 -         error = 0;
93 -         try
94 -             path_file_inds = strcat(string(app.selectedPath), '\', string(val), '\file_inds.mat');
95 -             d = load(path_file_inds);
96 -             error = 0;
97 -         catch err
98 -             %errorDlg(err.message)
99 -             error = 1;
100 -        end
101 -        if error == 0
102 -            days(end+1) = b(element);
103 -        end
104 -    end
105 -
106 - % Sort days in ascending order
107 -     dates = datetime(days, 'InputFormat', 'MM-dd-yy', 'Format', 'MM-dd-yy');
108 -     dates_ord = sort(dates);
109 -
110 - % Establish days shown in the ChooseDaysListBox
111 -     app.ChooseDaysListBox.Items = string(dates_ord);
112 -
113 - % Establish phases shown in ChoosePhaseListBox
114 -     app.phases = {'Control'; 'Perturbation'; 'Washout'};
115 -     app.ChoosePhaseListBox.Items = app.phases;
116 -
117 -
118 -
119 -     end

```

Callback functions

```

120 -
121 - % Value changed function: ChooseDaysListBox
122 - function ChooseDaysListBoxValueChanged(app, event)
123 -     % Establish app.day as the item selected and enable the user to
124 -     % choose more than one item
125 -     app.day = app.ChooseDaysListBox.Value;
126 -     app.ChooseDaysListBox.Multiselect = 'on';
127 - end
128 -
129 - % Button pushed function: SelectDayButton
130 - function SelectDayButtonPushed(app, event)
131 -     add_day = app.day;           % Day selected
132 -     app.total_days = [app.total_days; add_day]; % List of days selected
133 -     app.total_days = unique(app.total_days); % To avoid repeated days
134 -
135 -     app.SelectedDaysListBox.Items = app.total_days; % Establish selected days as items
136 -
137 - % Enable the user to visualize the next panel and the delete
138 - % buttons
139 -     app.SelectedDaysListBox.Visible = 'on';
140 -     app.SelectedDaysListBoxLabel.Visible = 'on';
141 -     app.DeleteDayButton.Visible = 'on';
142 -     app.DeleteAllDaysButton.Visible = 'on';
143 -     app.PhasePanel.Visible = 'on';
144 -
145 -     end

```

```

147 % Button pushed function: SelectAllDaysButton
148 function SelectAllDaysButtonPushed(app, event)
149
150 % Establish as SelectedDaysListBox items all the available days
151 % If it is empty, add all the days
152 if isempty(app.total_days)
153     app.total_days = [app.ChooseDaysListBox.Items];
154     app.SelectedDaysListBox.Items = app.total_days;
155
156 % If not, delete all items and add all the days to avoid
157 % repetitions
158 else
159     app.total_days = [];
160     app.total_days = [app.ChooseDaysListBox.Items];
161     app.SelectedDaysListBox.Items = app.total_days;
162 end
163
164 % Enable the user to visualize the next panel and the delete
165 % buttons
166 app.SelectedDaysListBox.Items = app.total_days;
167 app.SelectedDaysListBox.Visible = 'on';
168 app.SelectedDaysListBoxLabel.Visible = 'on';
169 app.DeleteDayButton.Visible = 'on';
170 app.DeleteAllDaysButton.Visible = 'on';
171 app.PhasePanel.Visible = 'on';
172
173 end
174
175 % Value changed function: SelectedDaysListBox
176 function SelectedDaysListBoxValueChanged(app, event)
177
178 % Establish app.value_day as the item that is being selected
179 app.value_day = app.SelectedDaysListBox.Value;
180
181 end
182
183 % Button pushed function: DeleteDayButton
184 function DeleteDayButtonPushed(app, event)
185
186 % Delete the selected item by eliminating it from the vector
187 % using the index and update the SelectedDaysListBox
188 [~, index] = ismember(app.value_day, app.total_days);
189 app.total_days(index) = [];
190 app.SelectedDaysListBox.Items = app.total_days;
191
192 end
193
194 % Button pushed function: DeleteAllDaysButton
195 function DeleteAllDaysButtonPushed(app, event)
196
197 % Delete all days from the vector and all the items
198 app.total_days = [];
199 app.SelectedDaysListBox.Items = {};
200
201 end
202
203 % Value changed function: ChoosePhaseListBox
204 function ChoosePhaseListBoxValueChanged(app, event)
205
206 % Establish app.phase_selected as the item selected and disable
207 % the multiselect option
208 app.phase_selected = app.ChoosePhaseListBox.Value;
209 app.ChoosePhaseListBox.Multiselect = 'off';
210
211 end
212
213 % Button pushed function: SelectPhaseButton
214 function SelectPhaseButtonPushed(app, event)
215
216 % Show the selected phase
217 app.SelectedPhaseListBox.Items = string(app.phase_selected);
218
219 % Enable the user to visualize the next panel
220 app.SelectedPhaseListBox.Visible = 'on';
221 app.SelectedPhaseListBoxLabel.Visible = 'on';
222 app.ShowTargetCellOptionsButton.Visible = 'on';
223 app.TargetCellPanel.Visible = 'on';
224
225 end

```

```

227 % Button pushed function: ShowTargetCellOptionsButton
228 function ShowTargetCellOptionsButtonPushed(app, event)
229
230 % Description of the variables
231 end_fname = '.CenterOut.mat';
232 days = app.total_days;
233 day_s = 1;
234 day_e = length(days);
235
236 % Extract the possible target cells for each day
237 for day_i = day_s:day_e
238     path_file_inds = strcat(string(app.selectedPath), '\', string(days(day_i)), '\file_inds.mat');
239     d = load(path_file_inds);
240     Const = fieldnames(d);
241     if strcmp(app.phase_selected, 'Control')
242         path = d.(Const{1});
243         path_length = length(path);
244     elseif strcmp(app.phase_selected, 'Perturbation')
245         path = d.(Const{2});
246         path_length = length(path);
247     else
248         path = d.(Const{3});
249         path_length = length(path);
250     end
251
252     for file = 2:path_length
253         try
254             if isempty(path)
255                 else
256                     app.id = num2str(path(file));
257                     if numel(app.id) == 4
258                         start_fname = 'Arthur.BC.0';
259                     else
260                         start_fname = 'Arthur.BC.';
261                     end
262                     path_file = strcat(string(app.selectedPath), '\', string(days(day_i)), '\', start_fname, num2str(path(file)), end_fname);
263                     e = load(path_file);
264                     const1 = fieldnames(e); % Variables that are included in this file
265                     struct_spikes = e.(const1{6}); % Spikes variable
266                     t_cells = fieldnames(struct_spikes); % List of all the units
267                     app.C(end+1) = t_cells(1:end-1);
268                 end
269             catch err
270                 %errordlg(err.message)
271             end
272         end
273     end
274
275 % Find the common values in all the files of all the days selected
276 uvals = app.C{1};
277 for iarr = 1:numel(app.C)
278     uvals = intersect(app.C{iarr}, uvals); %Vector with the common target_cells
279 end
280
281 % Delete the items that end with different value than 1 (we only want the
282 % ones ending with _1
283 app.target_cells_list = [];
284 for i = 1:length(uvals)
285     item = char(uvals(i));
286     if str2double(item(end)) == 1
287         app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
288     else
289     end
290 end
291
292 % Use the definitive target cells list as items for the
293 % ChooseTargetCellListBox and enable the visualization of the
294 % SelectTCButton and the number of TC available
295 app.ChooseTargetCellListBox.Items = app.target_cells_list;
296 app.ChooseTargetCellListBox.Visible = 'on';
297 app.ChooseTargetCellListBox.Label.Visible = 'on';
298 app.SelectTCButton.Visible = 'on';
299 app.NumTCAvailableTextArea.Visible = 'on';
300 app.NumTCAvailableTextArea.Label.Visible = 'on';
301 app.NumTCAvailableTextArea.Value = string(length(app.target_cells_list));
302
303 end
304
305 % Value changed function: ChooseTargetCellListBox
306 function ChooseTargetCellListBoxValueChanged(app, event)
307
308 % Stablish app.target_cell as the selected target cell and
309 % disable the Multiselect option
310 app.target_cell = app.ChooseTargetCellListBox.Value;
311 app.ChooseTargetCellListBox.Multiselect = 'off';
312
313 end

```

```

315 % Button pushed function: SelectTCButton
316 function SelectTCButtonPushed(app, event)
317
318 % Show the selected TC
319 app.SelectedTCTextArea.Visible = 'on';
320 app.SelectedTCTextAreaLabel.Visible = 'on';
321 app.SelectedTCTextArea.Value = app.target_cell;
322
323 % Description of the variables
324 path = app.selectedPath;
325 days = app.total_days;
326 phase_sel = app.phase_selected;
327 end_fname = '.CenterOut.mat';
328 target_cell = app.target_cell;
329 day_s = 1;
330 day_e = length(days);
331 num_days = day_e - day_s + 1;
332 panel_gris = uipanel(app.TrajectoriesWindow, 'Position', [0, 0, 1536, 600], 'Scrollable', 'on'); % To cover the existing curves if that is the case
333 row_panel = 0;
334
335 % Obtain and plot the trajectories
336 for day_i = day_s:day_e
337 % Create a panel per day and establish its dimensions in
338 % each case
339 if num_days <= 8
340 wh = (1536 - ((num_days + 1) * 20) * (1 / (num_days + 0.5) ^ (num_days - 1)));
341 panel = uipanel(app.TrajectoriesWindow, 'Position', [((1536 - wh * num_days) / (num_days + 1)) * day_i, wh * (day_i - 1), 20, wh, 580], 'Scrollable', 'on');
342 panel.Title = string(days(day_i));
343 panel.FontSize = 14;
344 panel.FontWeight = 'Bold';
345 else
346 if mod(num_days, 2) == 0
347 else
348 num_days = num_days + 1;
349 end
350 if mod((day_i - 1), num_days / 2) == 0
351 row_panel = row_panel + 1;
352 end
353
354 wh = (1536 - 20 * ((num_days / 2) + 1)) / (num_days / 2);
355 panel = uipanel(app.TrajectoriesWindow, 'Position', [20 * (day_i - (num_days / 2) * (row_panel - 1)) + wh * (day_i - (num_days / 2) * (row_panel - 1) - 1), ...
356 10 * (2 - row_panel + 1) * 2 + 265 * (2 - row_panel), wh, 265], 'Scrollable', 'on');
357 panel.Title = string(days(day_i));
358 panel.FontSize = 14;
359 panel.FontWeight = 'Bold';
360 end
361
362 path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
363 d = load(path_file_inds);
364 Const = fieldnames(d);
365 if strcmp(phase_sel, 'Control')
366 app.path_phase = d.(Const{1}); % ID number of the control files
367 path_length = length(app.path_phase);
368 elseif strcmp(phase_sel, 'Perturbation')
369 app.path_phase = d.(Const{2}); % ID number of the perturbation files
370 path_length = length(app.path_phase);
371 else
372 app.path_phase = d.(Const{3}); % ID number of the washout files
373 path_length = length(app.path_phase);
374 end
375
376 row = 0;
377 for file = 1:path_length
378 app.targs = [];
379 app.traj = [];
380 try
381 ax = uiaxes(panel);
382 title(ax, string(app.path_phase(file)));
383 if num_days == 1
384 if mod(path_length, 4) == 0
385 n_rows = path_length / 4;
386 else
387 n_rows = fix(path_length / 4) + 1;
388 end
389 if mod((file - 1), 4) == 0
390 row = row + 1;
391 end
392 wh_axis = (wh - 100) / 4;
393 space = (wh - 4 * wh_axis) / 5;
394 ax.Position = [space * (file - 4 * (row - 1)) + wh_axis * (file - 4 * (row - 1) - 1), wh_axis * (n_rows - row + 1) + 10, wh_axis, wh_axis];
395 elseif num_days > 1 && num_days <= 3
396 if mod(path_length, 3) == 0
397 n_rows = path_length / 3;
398 else
399 n_rows = fix(path_length / 3) + 1;
400 end
401 if mod((file - 1), 3) == 0
402 row = row + 1;
403 end
404 wh_axis = (wh - 100 * 0.5 ^ (num_days - 2)) / 3;
405 space = (wh - 3 * wh_axis) / 4 - 5;
406 ax.Position = [space * (file - 3 * (row - 1)) + wh_axis * (file - 3 * (row - 1) - 1), wh_axis * (n_rows - row + 1) + 10, wh_axis, wh_axis];
407 elseif num_days > 3 && num_days <= 5
408 if mod(path_length, 2) == 0
409 n_rows = path_length / 2;
410 else
411 n_rows = fix(path_length / 2) + 1;
412 end
413 if mod((file - 1), 2) == 0
414 row = row + 1;
415 end
416 wh_axis = (wh - 60 * 0.5 ^ (num_days - 4)) / 2;
417 space = (wh - 2 * wh_axis) / 3 - 5;
418 ax.Position = [space * (file - 2 * (row - 1)) + wh_axis * (file - 2 * (row - 1) - 1), wh_axis * (n_rows - row + 1) + 10, wh_axis, wh_axis];
419 elseif num_days > 5 && num_days <= 8
420 ratio = (1536 - ((num_days + 1) * 20) * (1 / (num_days + 0.5) ^ (num_days - 1))) / (1536 - ((num_days - 1) * 20) * (1 / (num_days - 1 + 0.5) ^ (num_days - 3)));

```



```

421 ~         space = 5*ratio;
422 ~         wh_axis = wh-2*space-1000*ratio;
423 ~         ax.Position = [space, wh_axis*(path_length-file)+10,wh_axis,wh_axis];
424 ~     else
425 ~         ratio = (1536-((num_days/2)+1)*20)^(1/(num_days/2+(0.5)^(num_days/2-1)))/(1536-((num_days/2-1)+1)*20)^(1/(num_days/2-1+(0.5)^(num_days/2-3)));
426 ~         space = 5*ratio;
427 ~         wh_axis = wh-2*space-1000*ratio;
428 ~         ax.Position = [space, wh_axis*(path_length-file)+10,wh_axis,wh_axis];
429 ~     end
430 ~
431 ~ % Obtain and plot the trajectories and the targets
432 ~ if isempty(app.path_phase)
433 ~ else
434 ~     x = num2str(app.path_phase(file));
435 ~     if numel(x) == 4
436 ~         start_fname = 'Arthur.BC.0';
437 ~     else
438 ~         start_fname = 'Arthur.BC.';
439 ~     end
440 ~     path_file = strcat(string(path), '\', string(days(day_i)), '\', start_fname, num2str(app.path_phase(file)), end_fname);
441 ~     try
442 ~         e = load(path_file);
443 ~     catch err
444 ~         %erroridg(err.message)
445 ~         continue
446 ~     end
447 ~
448 ~     discardedtrials=[];
449 ~     RT=150/1000;
450 ~     ndim=2;
451 ~
452 ~
453 ~ % Get zero positions of the cursor.
454 ~ zeroinds=find(e.em_feedback.CursorPosition(:,1)==0 & e.em_feedback.CursorPosition(:,2)==0 & e.em_feedback.CursorPosition(:,3)==0);
455 ~
456 ~ for i=1:length(e.trials.ComputerTrialTime)
457 ~     app.targs(i,:)=e.trials.TargetPos(i,[1:ndim]);
458 ~     poszeroinds=find(e.em_feedback.Time(zeroinds)>e.trials.ComputerTrialTime(i));
459 ~     emfbegind=zeroinds(poszeroinds(1));
460 ~     if length(poszeroinds)>1
461 ~         emfendind=zeroinds(poszeroinds(2))-1;
462 ~     else
463 ~         emfendind=length(e.em_feedback.Time);
464 ~     end
465 ~     emfinds=[emfbegind:emfendind];
466 ~     trajtimes=e.em_feedback.Time(emfinds)-e.trials.ComputerTrialTime(i);
467 ~     firsttrajtime=e.em_feedback.Time(emfinds(1));
468 ~     sns=e.em_feedback.SerialNo(emfinds);
469 ~     dsns=diff(sns);
470 ~     outoforderind=find(dsns<-1,1);
471 ~
472 ~     if isempty(outoforderind)
473 ~         % Get the spike count inds corresponding to these serial
474 ~         % numbers:
475 ~         try
476 ~             spcbegind=find(e.spike_counts.SerialNo==sns(1));
477 ~             spcendind=find(e.spike_counts.SerialNo==sns(end));
478 ~         catch
479 ~             spcbegind=find(e.spike_counts.headers.SerialNo==sns(1));
480 ~             spcendind=find(e.spike_counts.headers.SerialNo==sns(end));
481 ~         end
482 ~         spcinds=[spcbegind:spcendind];
483 ~
484 ~         % Get the em_movement_command inds corresponding to this range
485 ~         % of serial numbers:
486 ~         emcbegind=find(e.em_movement_command.SerialNo==sns(1));
487 ~         emcendind=find(e.em_movement_command.SerialNo==sns(end));
488 ~         emcinds=[emcbegind:emcendind];
489 ~         app.traj{i}.t=trajtimes;
490 ~         app.traj{i}.firststetime=firsttrajtime;
491 ~         app.traj{i}.computerstarttime=e.trials.ComputerTrialTime(i);
492 ~         app.traj{i}.HoldAStart=e.trials.HoldAStart(i);
493 ~         app.traj{i}.HoldAFinish=e.trials.HoldAFinish(i);
494 ~         app.traj{i}.HoldBStart=e.trials.HoldBStart(i);
495 ~         app.traj{i}.HoldBFinish=e.trials.HoldBFinish(i);
496 ~         app.traj{i}.emcomm_vel=e.em_movement_command.Velocity(emcinds,[1:ndim]);
497 ~         app.traj{i}.emfeed_pos=e.em_feedback.CursorPosition(emfinds,[1:ndim]);
498 ~         app.traj{i}.pos=app.traj{i}.emfeed_pos;
499 ~         app.traj{i}.InvisExit=nan(1,ndim);
500 ~         app.traj{i}.InvisExitTime=NaN;

```

```

581 ~ app.traj{i}.ReactionTime=e.trials.HoldAFinish(i)+RT;
582 ~ app.traj{i}.ReactionTimePos=interp1(app.traj{i}.t,app.traj{i}.pos,app.traj{i}.ReactionTime);
583 ~
584 ~
585 ~ if isfield(e.header.cout,'cursorInvisibleZone')
586 ~     r=e.header.cout.cursorInvisibleZone;
587 ~     dist=sqrt(sum(app.traj{i}.emfeed_pos.^2,2));
588 ~     outInds=find(dist>r);
589 ~     if ~isempty(outInds)
590 ~         app.traj{i}.InvisExit=app.traj{i}.emfeed_pos(outInds(1),:);
591 ~         app.traj{i}.InvisExitTime=app.traj{i}.t(outInds(1));
592 ~     end
593 ~ end
594 ~ else
595 ~     discardedTrials(end+1)=i;
596 ~     app.traj{i}.t=zeros(0,1);
597 ~     app.traj{i}.firtemtime=zeros(0,1);
598 ~     app.traj{i}.computerstarttime=zeros(0,1);
599 ~     app.traj{i}.HoldAStart=zeros(0,1);
600 ~     app.traj{i}.HoldAFinish=zeros(0,1);
601 ~     app.traj{i}.HoldBStart=zeros(0,1);
602 ~     app.traj{i}.HoldBFinish=zeros(0,1);
603 ~     app.traj{i}.InvisExit=NaN*zeros(0,ndim);
604 ~     app.traj{i}.InvisExitTime=NaN*zeros(0,1);
605 ~     app.traj{i}.ReactionTime=NaN*zeros(0,1);
606 ~     app.traj{i}.ReactionTimePos=NaN*zeros(0,ndim);
607 ~     app.traj{i}.emcomm_vel=zeros(0,ndim);
608 ~     app.traj{i}.emcomm_pos=zeros(0,ndim);
609 ~     app.traj{i}.emfeed_pos=zeros(0,ndim);
610 ~     app.traj{i}.pos=zeros(0,ndim);
611 ~ end
612 ~ for element = 1:length(app.traj)
613 ~     hold(ax,"on")
614 ~     plot(ax, app.traj{element}.pos(:,1),app.traj{element}.pos(:,2))
615 ~     plot(ax, app.targs(:,1),app.targs(:,2),'ko')
616 ~     hold(ax, "off")
617 ~ end
618 ~ end
619 ~ catch err
620 ~     %errordlg(err.message)
621 ~ end
622 ~ end
623 ~ end
624 ~ end
625 ~ end
626 ~ end
627 ~ end
628 ~ end
629 ~ end
630 ~ end
631 ~ end
632 ~ end
633 ~ end
634 ~ end
635 ~ end
636 ~ end
637 ~ end
638 ~ end
639 ~ end
640 ~ end
641 ~ end
642 ~ end
643 ~ end
644 ~ end
645 ~ end
646 ~ end
647 ~ end
648 ~ end
649 ~ end
650 ~ end
651 ~ end
652 ~ end
653 ~ end
654 ~ end
655 ~ end
656 ~ end
657 ~ end
658 ~ end
659 ~ end
660 ~ end
661 ~ end
662 ~ end
663 ~ end
664 ~ end
665 ~ end
666 ~ end
667 ~ end
668 ~ end
669 ~ end
670 ~ end
671 ~ end
672 ~ end
673 ~ end
674 ~ end
675 ~ end
676 ~ end
677 ~ end
678 ~ end
679 ~ end
680 ~ end
681 ~ end
682 ~ end
683 ~ end
684 ~ end
685 ~ end
686 ~ end
687 ~ end
688 ~ end
689 ~ end
690 ~ end
691 ~ end
692 ~ end
693 ~ end
694 ~ end
695 ~ end
696 ~ end
697 ~ end
698 ~ end
699 ~ end
700 ~ end
701 ~ end
702 ~ end
703 ~ end
704 ~ end
705 ~ end
706 ~ end
707 ~ end
708 ~ end
709 ~ end
710 ~ end
711 ~ end
712 ~ end
713 ~ end
714 ~ end
715 ~ end
716 ~ end
717 ~ end
718 ~ end
719 ~ end
720 ~ end
721 ~ end
722 ~ end
723 ~ end
724 ~ end
725 ~ end
726 ~ end
727 ~ end
728 ~ end
729 ~ end
730 ~ end
731 ~ end
732 ~ end
733 ~ end
734 ~ end
735 ~ end
736 ~ end
737 ~ end
738 ~ end
739 ~ end
740 ~ end
741 ~ end
742 ~ end
743 ~ end
744 ~ end
745 ~ end
746 ~ end
747 ~ end
748 ~ end
749 ~ end
750 ~ end
751 ~ end
752 ~ end
753 ~ end
754 ~ end
755 ~ end
756 ~ end
757 ~ end
758 ~ end
759 ~ end
760 ~ end
761 ~ end
762 ~ end
763 ~ end
764 ~ end
765 ~ end
766 ~ end
767 ~ end
768 ~ end
769 ~ end
770 ~ end
771 ~ end
772 ~ end
773 ~ end
774 ~ end
775 ~ end
776 ~ end
777 ~ end
778 ~ end
779 ~ end
780 ~ end
781 ~ end
782 ~ end
783 ~ end
784 ~ end
785 ~ end
786 ~ end
787 ~ end
788 ~ end
789 ~ end
790 ~ end
791 ~ end
792 ~ end
793 ~ end
794 ~ end
795 ~ end
796 ~ end
797 ~ end
798 ~ end
799 ~ end
800 ~ end
801 ~ end
802 ~ end
803 ~ end
804 ~ end
805 ~ end
806 ~ end
807 ~ end
808 ~ end
809 ~ end
810 ~ end
811 ~ end
812 ~ end
813 ~ end
814 ~ end
815 ~ end
816 ~ end
817 ~ end
818 ~ end
819 ~ end
820 ~ end
821 ~ end
822 ~ end
823 ~ end
824 ~ end
825 ~ end
826 ~ end
827 ~ end
828 ~ end
829 ~ end
830 ~ end
831 ~ end
832 ~ end
833 ~ end
834 ~ end
835 ~ end
836 ~ end
837 ~ end
838 ~ end
839 ~ end
840 ~ end
841 ~ end
842 ~ end
843 ~ end
844 ~ end
845 ~ end
846 ~ end
847 ~ end
848 ~ end
849 ~ end
850 ~ end
851 ~ end
852 ~ end
853 ~ end
854 ~ end
855 ~ end
856 ~ end
857 ~ end
858 ~ end
859 ~ end
860 ~ end
861 ~ end
862 ~ end
863 ~ end
864 ~ end
865 ~ end
866 ~ end
867 ~ end
868 ~ end
869 ~ end
870 ~ end
871 ~ end
872 ~ end
873 ~ end
874 ~ end
875 ~ end
876 ~ end
877 ~ end
878 ~ end
879 ~ end
880 ~ end
881 ~ end
882 ~ end
883 ~ end
884 ~ end
885 ~ end
886 ~ end
887 ~ end
888 ~ end
889 ~ end
890 ~ end
891 ~ end
892 ~ end
893 ~ end
894 ~ end
895 ~ end
896 ~ end
897 ~ end
898 ~ end
899 ~ end
900 ~ end
901 ~ end
902 ~ end
903 ~ end
904 ~ end
905 ~ end
906 ~ end
907 ~ end
908 ~ end
909 ~ end
910 ~ end
911 ~ end
912 ~ end
913 ~ end
914 ~ end
915 ~ end
916 ~ end
917 ~ end
918 ~ end
919 ~ end
920 ~ end
921 ~ end
922 ~ end
923 ~ end
924 ~ end
925 ~ end
926 ~ end
927 ~ end
928 ~ end
929 ~ end
930 ~ end
931 ~ end
932 ~ end
933 ~ end
934 ~ end
935 ~ end
936 ~ end
937 ~ end
938 ~ end
939 ~ end
940 ~ end
941 ~ end
942 ~ end
943 ~ end
944 ~ end
945 ~ end
946 ~ end
947 ~ end
948 ~ end
949 ~ end
950 ~ end
951 ~ end
952 ~ end
953 ~ end
954 ~ end
955 ~ end
956 ~ end
957 ~ end
958 ~ end
959 ~ end
960 ~ end
961 ~ end
962 ~ end
963 ~ end
964 ~ end
965 ~ end
966 ~ end
967 ~ end
968 ~ end
969 ~ end
970 ~ end
971 ~ end
972 ~ end
973 ~ end
974 ~ end
975 ~ end
976 ~ end
977 ~ end
978 ~ end
979 ~ end
980 ~ end
981 ~ end
982 ~ end
983 ~ end
984 ~ end
985 ~ end
986 ~ end
987 ~ end
988 ~ end
989 ~ end
990 ~ end
991 ~ end
992 ~ end
993 ~ end
994 ~ end
995 ~ end
996 ~ end
997 ~ end
998 ~ end
999 ~ end
1000 ~ end

```

Component initialization

```

565 ~ % Component initialization
566 ~ methods (Access = private)
567 ~
568 ~ % Create UIFigure and components
569 ~ function createComponents(app)
570 ~
571 ~ % Create TrajectoriesWindow and hide until all components are created
572 ~ app.TrajectoriesWindow = uifigure('Visible', 'off');
573 ~ app.TrajectoriesWindow.Position = [0 40 1536 800];
574 ~ app.TrajectoriesWindow.Name = 'MATLAB App';
575 ~
576 ~ % Create TargetCellPanel
577 ~ app.TargetCellPanel = uipanel(app.TrajectoriesWindow);
578 ~ app.TargetCellPanel.TitlePosition = 'centertop';
579 ~ app.TargetCellPanel.Title = 'Target Cell';
580 ~ app.TargetCellPanel.BackgroundColor = [1 1 1];
581 ~ app.TargetCellPanel.Position = [1031 612 484 178];
582 ~
583 ~ % Create SelectedTCTextAreaLabel
584 ~ app.SelectedTCTextAreaLabel = uilabel(app.TargetCellPanel);
585 ~ app.SelectedTCTextAreaLabel.HorizontalAlignment = 'right';
586 ~ app.SelectedTCTextAreaLabel.Position = [274 78 75 22];
587 ~ app.SelectedTCTextAreaLabel.Text = 'Selected TC';
588 ~
589 ~ % Create SelectedTCTextArea
590 ~ app.SelectedTCTextArea = uitextarea(app.TargetCellPanel);
591 ~ app.SelectedTCTextArea.Editable = 'off';
592 ~ app.SelectedTCTextArea.WordWrap = 'off';
593 ~ app.SelectedTCTextArea.Position = [353 76 105 25];
594 ~
595 ~ % Create NumTCAvailableTextAreaLabel
596 ~ app.NumTCAvailableTextAreaLabel = uilabel(app.TargetCellPanel);
597 ~ app.NumTCAvailableTextAreaLabel.HorizontalAlignment = 'right';
598 ~ app.NumTCAvailableTextAreaLabel.Position = [274 114 105 22];
599 ~ app.NumTCAvailableTextAreaLabel.Text = 'Num. TC Available';
600 ~
601 ~ % Create NumTCAvailableTextArea
602 ~ app.NumTCAvailableTextArea = uitextarea(app.TargetCellPanel);
603 ~ app.NumTCAvailableTextArea.Editable = 'off';
604 ~ app.NumTCAvailableTextArea.Position = [387 113 33 24];

```

```

605
606 % Create SelectTCButton
607 app.SelectTCButton = uibutton(app.TargetCellPanel, 'push');
608 app.SelectTCButton.ButtonPushedFcn = createCallbackFcn(app, @SelectTCButtonPushed, true);
609 app.SelectTCButton.Position = [58 8 100 22];
610 app.SelectTCButton.Text = 'Select TC';
611
612 % Create ChooseTargetCellListBoxLabel
613 app.ChooseTargetCellListBoxLabel = uilabel(app.TargetCellPanel);
614 app.ChooseTargetCellListBoxLabel.Position = [47 105 111 22];
615 app.ChooseTargetCellListBoxLabel.Text = 'Choose Target Cell: ';
616
617 % Create ChooseTargetCellListBox
618 app.ChooseTargetCellListBox = uilistbox(app.TargetCellPanel);
619 app.ChooseTargetCellListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseTargetCellListBoxValueChanged, true);
620 app.ChooseTargetCellListBox.Position = [47 39 129 66];
621
622 % Create ShowTargetCellOptionsButton
623 app.ShowTargetCellOptionsButton = uibutton(app.TargetCellPanel, 'push');
624 app.ShowTargetCellOptionsButton.ButtonPushedFcn = createCallbackFcn(app, @ShowTargetCellOptionsButtonPushed, true);
625 app.ShowTargetCellOptionsButton.Position = [36 130 151 22];
626 app.ShowTargetCellOptionsButton.Text = 'Show Target Cell Options';
627
628 % Create SaveScreenshotButton
629 app.SaveScreenshotButton = uibutton(app.TargetCellPanel, 'state');
630 app.SaveScreenshotButton.ValueChangedFcn = createCallbackFcn(app, @SaveScreenshotButtonValueChanged, true);
631 app.SaveScreenshotButton.Text = 'Save Screenshot';
632 app.SaveScreenshotButton.FontWeight = 'bold';
633 app.SaveScreenshotButton.Position = [310 29 112 22];
634
635 % Create PhasePanel
636 app.PhasePanel = uipanel(app.TrajectoriesWindow);
637 app.PhasePanel.TitlePosition = 'centertop';
638 app.PhasePanel.Title = 'Phase';
639 app.PhasePanel.BackgroundColor = [1 1 1];
640 app.PhasePanel.Position = [526 612 484 178];
641
642 % Create SelectPhaseButton
643 app.SelectPhaseButton = uibutton(app.PhasePanel, 'push');
644 app.SelectPhaseButton.ButtonPushedFcn = createCallbackFcn(app, @SelectPhaseButtonPushed, true);
645 app.SelectPhaseButton.BackgroundColor = [1 1 1];
646 app.SelectPhaseButton.Position = [95 35 100 22];
647 app.SelectPhaseButton.Text = 'Select Phase';
648
649 % Create SelectedPhaseListBoxLabel
650 app.SelectedPhaseListBoxLabel = uilabel(app.PhasePanel);
651 app.SelectedPhaseListBoxLabel.Position = [289 136 93 22];
652 app.SelectedPhaseListBoxLabel.Text = 'Selected Phase: ';
653
654 % Create SelectedPhaseListBox
655 app.SelectedPhaseListBox = uilistbox(app.PhasePanel);
656 app.SelectedPhaseListBox.Position = [289 62 100 74];
657
658 % Create ChoosePhaseListBoxLabel
659 app.ChoosePhaseListBoxLabel = uilabel(app.PhasePanel);
660 app.ChoosePhaseListBoxLabel.Position = [95 136 87 22];
661 app.ChoosePhaseListBoxLabel.Text = 'Choose Phase: ';
662
663 % Create ChoosePhaseListBox
664 app.ChoosePhaseListBox = uilistbox(app.PhasePanel);
665 app.ChoosePhaseListBox.ValueChangedFcn = createCallbackFcn(app, @ChoosePhaseListBoxValueChanged, true);
666 app.ChoosePhaseListBox.Position = [95 63 100 74];
667
668 % Create DaysPanel
669 app.DaysPanel = uipanel(app.TrajectoriesWindow);
670 app.DaysPanel.TitlePosition = 'centertop';
671 app.DaysPanel.Title = 'Days';
672 app.DaysPanel.BackgroundColor = [1 1 1];
673 app.DaysPanel.Position = [21 612 484 178];
674
675 % Create DeleteAllDaysButton
676 app.DeleteAllDaysButton = uibutton(app.DaysPanel, 'push');
677 app.DeleteAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllDaysButtonPushed, true);
678 app.DeleteAllDaysButton.FontSize = 8;
679 app.DeleteAllDaysButton.Position = [352 37 71 20];
680 app.DeleteAllDaysButton.Text = 'Delete All Days';
681
682 % Create DeleteDayButton
683 app.DeleteDayButton = uibutton(app.DaysPanel, 'push');
684 app.DeleteDayButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteDayButtonPushed, true);
685 app.DeleteDayButton.FontSize = 8;
686 app.DeleteDayButton.Position = [287 36 51 20];
687 app.DeleteDayButton.Text = 'Delete Day';

```

```

688
689 % Create SelectAllDaysButton
690 app.SelectAllDaysButton = uibutton(app.DaysPanel, 'push');
691 app.SelectAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllDaysButtonPushed, true);
692 app.SelectAllDaysButton.FontSize = 8;
693 app.SelectAllDaysButton.Position = [98 8 67 22];
694 app.SelectAllDaysButton.Text = 'Select All Days';
695
696 % Create SelectDayButton
697 app.SelectDayButton = uibutton(app.DaysPanel, 'push');
698 app.SelectDayButton.ButtonPushedFcn = createCallbackFcn(app, @SelectDayButtonPushed, true);
699 app.SelectDayButton.BackgroundColor = [1 1 1];
700 app.SelectDayButton.Position = [82 36 100 22];
701 app.SelectDayButton.Text = 'Select Day';
702
703 % Create SelectedDaysListBoxLabel
704 app.SelectedDaysListBoxLabel = uilabel(app.DaysPanel);
705 app.SelectedDaysListBoxLabel.Position = [287 136 86 22];
706 app.SelectedDaysListBoxLabel.Text = 'Selected Days:';
707
708 % Create SelectedDaysListBox
709 app.SelectedDaysListBox = uilistbox(app.DaysPanel);
710 app.SelectedDaysListBox.Multiselect = 'on';
711 app.SelectedDaysListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedDaysListBoxValueChanged, true);
712 app.SelectedDaysListBox.Position = [287 63 136 72];
713 app.SelectedDaysListBox.Value = {'Item 1'};
714
715 % Create ChoosedaysListBoxLabel
716 app.ChoosedaysListBoxLabel = uilabel(app.DaysPanel);
717 app.ChoosedaysListBoxLabel.Position = [61 136 79 22];
718 app.ChoosedaysListBoxLabel.Text = 'Choose days:';
719
720 % Create ChooseDaysListBox
721 app.ChooseDaysListBox = uilistbox(app.DaysPanel);
722 app.ChooseDaysListBox.Multiselect = 'on';
723 app.ChooseDaysListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseDaysListBoxValueChanged, true);
724 app.ChooseDaysListBox.Position = [61 63 136 72];
725 app.ChooseDaysListBox.Value = {'Item 1'};
726
727 % Show the figure after all components are created
728 app.TrajectoriesWindow.Visible = 'on';
729
730 end
731
732 % App creation and deletion
733 methods (Access = public)
734
735 % Construct app
736 function app = Trajectories_FINAL(varargin)
737
738 % Create UIFigure and components
739 createComponents(app)
740
741 % Register the app with App Designer
742 registerApp(app, app.TrajectoriesWindow)
743
744 % Execute the startup function
745 runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
746
747 if nargin == 0
748 clear app
749 end
750
751 % Code that executes before app deletion
752 function delete(app)
753
754 % Delete UIFigure when app is deleted
755 delete(app.TrajectoriesWindow)
756
757 end
758
759 end

```

ANNEX IV. Code used for the development of the Control results window

Properties

```

1 classdef TFG_Sentences_Control < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5         UIFigure          matlab.ui.Figure
6         Control            matlab.ui.control.TextArea
7         PerturbationTextAreaLabel_2  matlab.ui.control.Label
8         SaveResultsButton  matlab.ui.control.Button
9     end
10

```

Startup function

```

14     % Code that executes after component creation
15     function startupFcn(app, all_sentences_control)
16
17         app.UIFigure.Name = 'CONTROL: Angle of preference & Spiking rate';
18         app.Control.Value = string(all_sentences_control);
19
20     end

```

Callback functions

```

21     % Button pushed function: SaveResultsButton
22     function SaveResultsButtonPushed(app, event)
23
24         T = cell2table(app.Control.Value(1:end,:)); % Convert cell array to table
25         [file,path] = uiputfile('','.csv'); % Open dialog box for saving files
26         filename = fullfile(path,file);
27         if ischar(filename) % If the name has the correct format, save the file
28             writetable(T,filename) % Write the table to a CSV file
29         end
30         app.UIFigure.Visible = 'off'; % These two lines of code work-around an issue whether the figure is sent to the background.
31         app.UIFigure.Visible = 'on';
32
33     end
34
35 end

```

Component initialization

```

39     % Create UIFigure and components
40     function createComponents(app)
41
42         % Create UIFigure and hide until all components are created
43         app.UIFigure = uifigure('Visible', 'off');
44         app.UIFigure.Position = [0 630 512 208];
45         app.UIFigure.Name = 'MATLAB App';
46         app.UIFigure.HandleVisibility = 'on';
47
48         % Create Control
49         app.Control = uicontrol(app.UIFigure);
50         app.Control.Position = [23 43 466 135];
51
52         % Create PerturbationTextAreaLabel_2
53         app.PerturbationTextAreaLabel_2 = uicontrol(app.UIFigure);
54         app.PerturbationTextAreaLabel_2.HorizontalAlignment = 'center';
55         app.PerturbationTextAreaLabel_2.FontSize = 14;
56         app.PerturbationTextAreaLabel_2.FontWeight = 'bold';
57         app.PerturbationTextAreaLabel_2.Position = [228 180 55 22];
58         app.PerturbationTextAreaLabel_2.Text = 'Control';
59
60         % Create SaveResultsButton
61         app.SaveResultsButton = uicontrol(app.UIFigure, 'push');
62         app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
63         app.SaveResultsButton.FontWeight = 'bold';
64         app.SaveResultsButton.Position = [206 10 100 22];
65         app.SaveResultsButton.Text = 'Save Results';
66
67         % Show the figure after all components are created
68         app.UIFigure.Visible = 'on';
69     end
70
71 end
72
73 % Construct app
74 function app = TFG_Sentences_Control(varargin)
75
76     % Create UIFigure and components
77     createComponents(app)
78
79     % Register the app with App Designer
80     registerApp(app, app.UIFigure)
81
82     % Execute the startup function
83     runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
84
85     if nargin == 0
86         clear app
87     end
88
89 end
90
91 % Code that executes before app deletion
92 function delete(app)
93
94     % Delete UIFigure when app is deleted
95     delete(app.UIFigure)
96
97 end
98
99 end
100

```

ANNEX V. Code used for the development of the Perturbation results window

Properties

```

1 classdef TFG_Sentences_Perturbation < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public)
5         UIFigure          matlab.ui.Figure
6         PerturbationTextAreaLabel  matlab.ui.control.Label
7         Perturbation      matlab.ui.control.TextArea
8         SaveResultsButton  matlab.ui.control.Button
9     end

```

Startup function

```

14 % Code that executes after component creation
15 function startupFcn(app, all_sentences_perturbation)
16
17     app.UIFigure.Name = 'PERTURBATION: Angle of preference & Spiking rate';
18     app.Perturbation.Value = string(all_sentences_perturbation);
19
20 end

```

Callback functions

```

22 % Button pushed function: SaveResultsButton
23 function SaveResultsButtonPushed(app, event)
24
25     T = cell2table(app.Perturbation.Value(1:end,:)); % Convert cell array to table
26     [file,path] = uiputfile('','*.csv'); % Open dialog box for saving files
27     filename = fullfile(path,file);
28     if ischar(filename) % If the name has the correct format, save the file
29         writetable(T,filename) % Write the table to a CSV file
30     end
31     app.UIFigure.Visible = 'off'; % These two lines of code work-around an issue whether the figure is sent to the background.
32     app.UIFigure.Visible = 'on';
33
34 end
35

```

Component initialization

```

39
40 % Create UIFigure and components
41 function createComponents(app)
42
43     % Create UIFigure and hide until all components are created
44     app.UIFigure = uifigure('Visible','off');
45     app.UIFigure.Position = [512 630 512 200];
46     app.UIFigure.Name = 'MATLAB App';
47     app.UIFigure.HandleVisibility = 'on';
48
49     % Create PerturbationTextAreaLabel
50     app.PerturbationTextAreaLabel = uilabel(app.UIFigure);
51     app.PerturbationTextAreaLabel.HorizontalAlignment = 'center';
52     app.PerturbationTextAreaLabel.FontSize = 14;
53     app.PerturbationTextAreaLabel.FontWeight = 'bold';
54     app.PerturbationTextAreaLabel.Position = [211 180 89 22];
55     app.PerturbationTextAreaLabel.Text = 'Perturbation';
56
57     % Create Perturbation
58     app.Perturbation = uitextarea(app.UIFigure);
59     app.Perturbation.Position = [23 43 466 135];
60
61     % Create SaveResultsButton
62     app.SaveResultsButton = uibutton(app.UIFigure, 'push');
63     app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
64     app.SaveResultsButton.FontWeight = 'bold';
65     app.SaveResultsButton.Position = [206 10 100 22];
66     app.SaveResultsButton.Text = 'Save Results';
67
68     % Show the figure after all components are created
69     app.UIFigure.Visible = 'on';
70
71 end
72
73 % Construct app
74 function app = TFG_Sentences_Perturbation(varargin)
75
76     % Create UIFigure and components
77     createComponents(app)
78
79     % Register the app with App Designer
80     registerApp(app, app.UIFigure)
81
82     % Execute the startup function
83     runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
84
85     if nargin == 0
86         clear app
87     end
88
89 end
90
91 % Code that executes before app deletion
92 function delete(app)
93
94     % Delete UIFigure when app is deleted
95     delete(app.UIFigure)
96
97 end
98
99 end
100

```

ANNEX VI. Code used for the development of the Washout results window

Properties

```

1 classdef TFG_Sentences_Washout < matlab.apps.AppBase
2
3 % Properties that correspond to app components
4 properties (Access = public)
5     UIFigure          matlab.ui.Figure
6     Washout           matlab.ui.control.TextArea
7     PerturbationTextAreaLabel_3  matlab.ui.control.Label
8     SaveResultsButton matlab.ui.control.Button
9 end

```

Startup function

```

14 % Code that executes after component creation
15 function startupFcn(app, all_sentences_washout)
16
17     app.UIFigure.Name = 'WASHOUT: Angle of preference & Spiking rate';
18     app.Washout.Value = string(all_sentences_washout);
19
20 end

```

Callback functions

```

22 % Button pushed function: SaveResultsButton
23 function SaveResultsButtonPushed(app, event)
24
25     T = cell2table(app.Washout.Value(1:end,:)); % Convert cell array to table
26     [file,path] = uiputfile('*.csv'); % Open dialog box for saving files
27     filename = fullfile(path,file);
28     if ischar(filename) % If the name has the correct format, save the file
29         writetable(T,filename) % Write the table to a CSV file
30     end
31     app.UIFigure.Visible = 'off'; % These two lines of code work-around an issue whether the figure is sent to the background.
32     app.UIFigure.Visible = 'on';
33
34 end
35

```

Component initialization

```

39 % Create UIFigure and components
40 function createComponents(app)
41
42 % Create UIFigure and hide until all components are created
43 app.UIFigure = uifigure('Visible', 'off');
44 app.UIFigure.Position = [1024 630 512 200];
45 app.UIFigure.Name = 'MATLAB App';
46 app.UIFigure.HandleVisibility = 'on';
47
48 % Create Washout
49 app.Washout = uitextarea(app.UIFigure);
50 app.Washout.Position = [23 43 466 135];
51
52 % Create PerturbationTextAreaLabel_3
53 app.PerturbationTextAreaLabel_3 = uilabel(app.UIFigure);
54 app.PerturbationTextAreaLabel_3.HorizontalAlignment = 'center';
55 app.PerturbationTextAreaLabel_3.FontSize = 14;
56 app.PerturbationTextAreaLabel_3.FontWeight = 'bold';
57 app.PerturbationTextAreaLabel_3.Position = [225 180 64 22];
58 app.PerturbationTextAreaLabel_3.Text = 'Washout';
59
60 % Create SaveResultsButton
61 app.SaveResultsButton = uibutton(app.UIFigure, 'push');
62 app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
63 app.SaveResultsButton.FontWeight = 'bold';
64 app.SaveResultsButton.Position = [206 10 100 22];
65 app.SaveResultsButton.Text = 'Save Results';
66
67 % Show the figure after all components are created
68 app.UIFigure.Visible = 'on';
69
70 end
71
72 % Construct app
73 function app = TFG_Sentences_Washout(varargin)
74
75 % Create UIFigure and components
76 createComponents(app)
77
78 % Register the app with App Designer
79 registerApp(app, app.UIFigure)
80
81 % Execute the startup function
82 runStartupFcn(app, @(app)startupFcn(app, varargin{:}))
83
84 if nargin == 0
85     clear app
86 end
87
88 % Code that executes before app deletion
89 function delete(app)
90
91 % Delete UIFigure when app is deleted
92 delete(app.UIFigure)
93
94 end
95
96 end
97
98 end
99
100

```

ANNEX VII. List of Figures and Tables

List of figures

FIGURE 1. SCHEMATIC OF A BCI SYSTEM.....	17
FIGURE 2. INTUITIVE NEURAL REPERTOIRE OBTAINED WHILE USING AN INTUITIVE MAPPING.	19
FIGURE 3. CONTENTS OF A FILE_INDS.MAT FILE	21
FIGURE 4. BUTTON COMPONENT	25
FIGURE 5. LIST BOX COMPONENT.....	26
FIGURE 6. PANEL COMPONENT.....	27
FIGURE 7. MAIN WINDOW FROM THE FIRST VERSION.....	28
FIGURE 8. TUNING CURVES WINDOW FROM THE FIRST VERSION.....	29
FIGURE 9. TRAJECTORIES WINDOW FROM THE FIRST VERSION	30
FIGURE 10. MAIN WINDOW.....	32
FIGURE 11. MAIN WINDOW COMPONENTS	32
FIGURE 12. MAIN WINDOW PROPERTIES	33
FIGURE 13. MAIN WINDOW STARTUP FUNCTION	33
FIGURE 14. BROWSE BUTTON CALLBACK FUNCTION	34
FIGURE 15. VISUALIZE TUNING CURVES BUTTON AND VISUALIZE TRAJECTORIES BUTTON CALLBACK FUNCTIONS.....	34
FIGURE 16. TUNING CURVES WINDOW.....	35
FIGURE 17. TUNING CURVES WINDOW COMPONENTS	36
FIGURE 18. INDEPENDENT WINDOWS SHOWING THE ANGLE OF PREFERENCE AND ITS SPIKING RATE	36
FIGURE 19. CODE USED TO OBTAIN THE COMMON CELLS LIST	38
FIGURE 20. GENERATED CODE TO OBTAIN THE FINAL TARGET CELL LIST.....	38
FIGURE 21. CODE USED TO GENERATE AND SAVE THE SCREENSHOT.....	39
FIGURE 22. TRAJECTORIES WINDOW	40
FIGURE 23. WORK BREAKDOWN STRUCTURE	45
FIGURE 24. PERT DIAGRAM	46

List of tables

TABLE 1. SOFTWARE COMPARISON	14
TABLE 2. FINAL SOFTWARE COMPARISON	14
TABLE 3. SWOT ANALYSIS	44
TABLE 4. SEQUENCE AND TIME MATRIX OF THE ACTIVITIES.....	46
TABLE 5. GANTT DIAGRAM	47