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

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 (EEC). 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 Mattew 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.

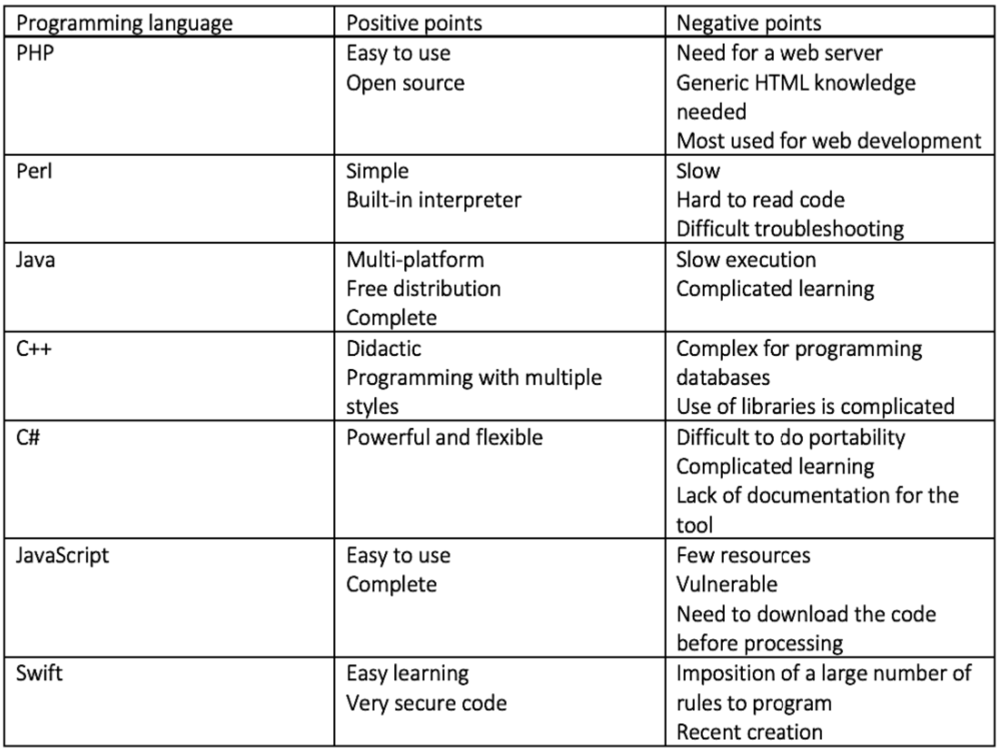
As for the prospects for new applications and new products related to BCI technology, it has been predicted that the first computers capable of simulating a human brain will emerge between now and 2045. This will enable the transplantation of human brains into robots and the complete understanding of 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' 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.

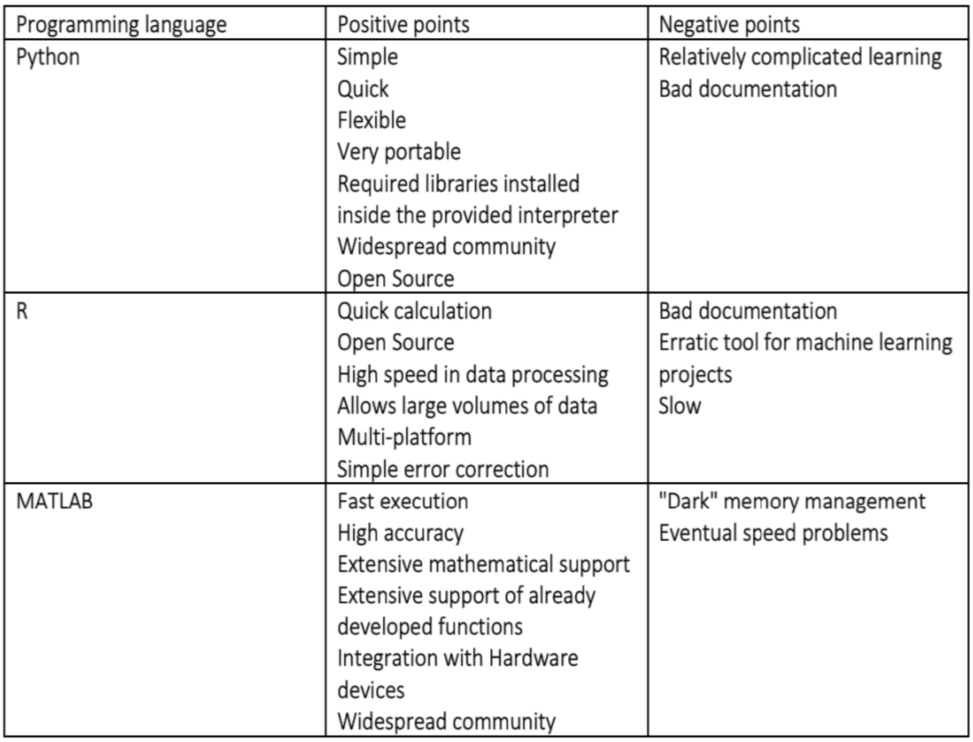
In the table below, the programming languages that have been discarded after the first exhaustive study of the different programming languages that are used today by programmers can be seen. Despite the fact that all seven of these languages have significant positive points, they are characterized by 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.

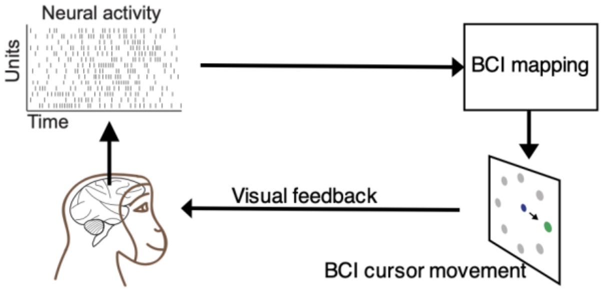


Figure 1. Schematic of a BCI system[1]

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 stablished 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]

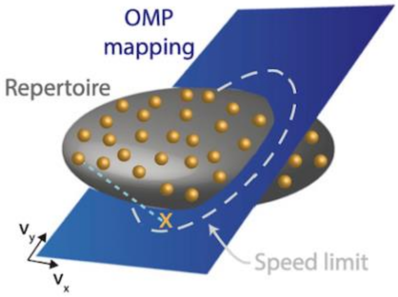


Figure 2. Intuitive neural repertoire obtained while using an intuitive mapping.[2]

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.

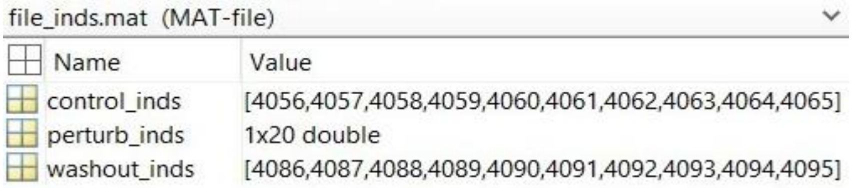
5.2. Graphic interface development

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.



file_inds.mat (MAT-file)

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/bindwidth, 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

MATLAB[3], 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.

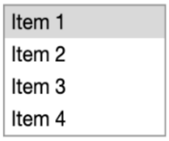


Figure 5. List Box component

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]

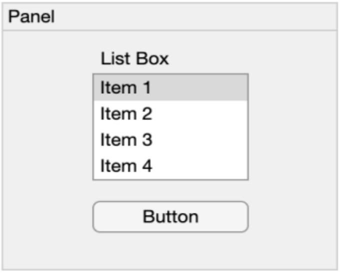


Figure 6. Panel component

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.

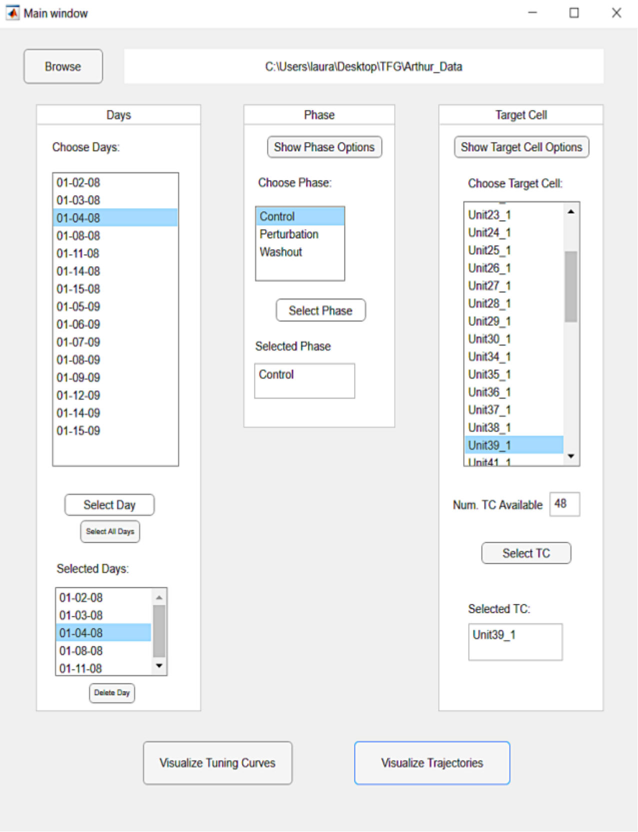
Finally, it should be noted that other elements have been used such as the Table component, which has allowed to visualize the outputs with matrix or vector structure of the functions used and mentioned in section 5.2.2., allowing to compare the results obtained by both in MATLAB and App Designer. This 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.



C:\Users\laura\Desktop\TFG\Arthur_Data

Choose Days:
01-02-08
01-03-08
01-04-08
01-08-08
01-11-08
01-14-08
01-15-08
01-05-09
01-06-09
01-07-09
01-08-09
01-09-09
01-12-09
01-14-09
01-15-09

Selected Days:
01-02-08
01-03-08
01-04-08
01-08-08
01-11-08

Choose Phase:
Control
Perturbation
Washout

Selected Phase
Control

Choose Target Cell:
Unit23_1
Unit24_1
Unit25_1
Unit26_1
Unit27_1
Unit28_1
Unit29_1
Unit30_1
Unit34_1
Unit35_1
Unit36_1
Unit37_1
Unit38_1
Unit39_1
Unit41_1

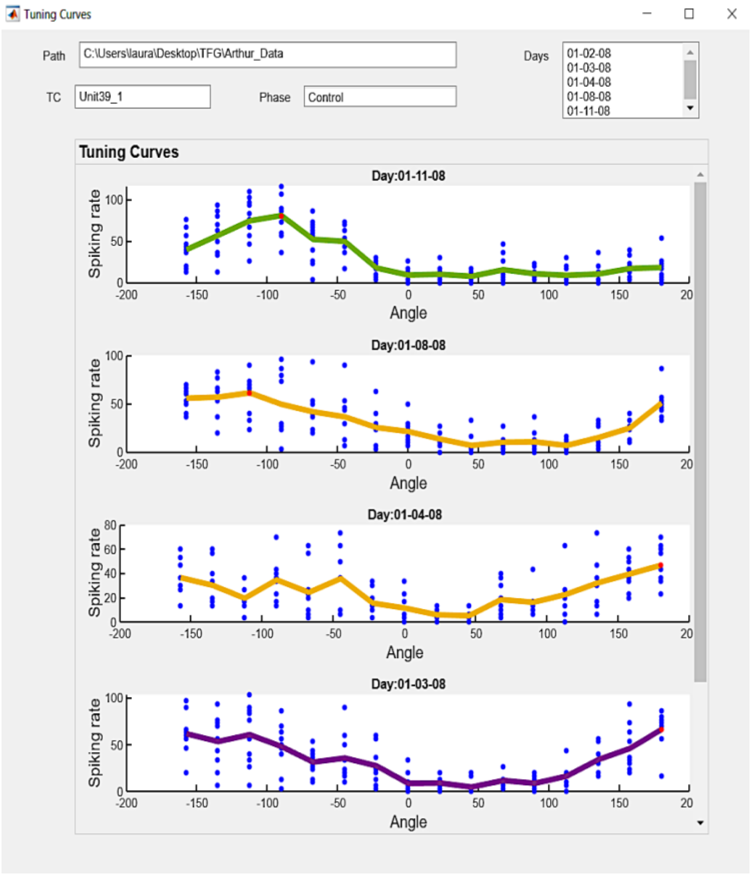
Num. TC Available 48

Selected TC:
Unit39_1

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.



Tuning Curves

Path: C:\Users\laura\Desktop\TFG\Arthur_Data

TC: Unit39_1

Phase: Control

Days:
01-02-08
01-03-08
01-04-08
01-08-08
01-11-08

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.

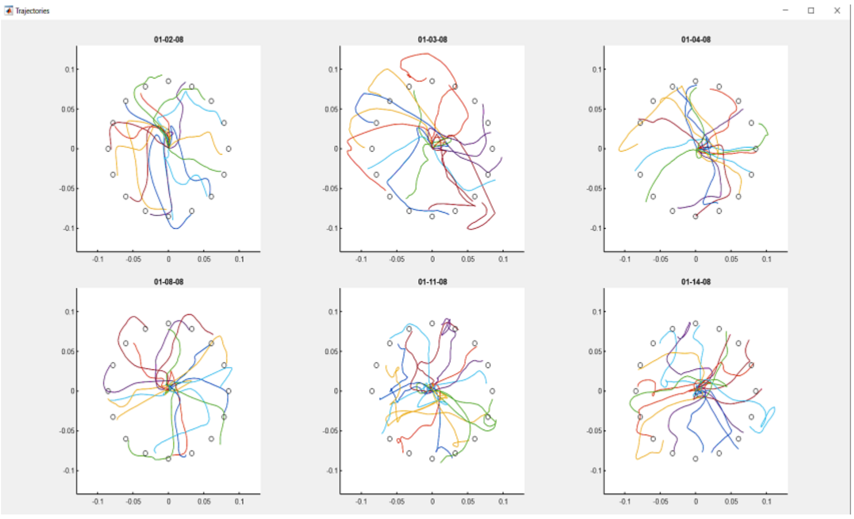


Figure 9. Trajectories window from the first version

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.

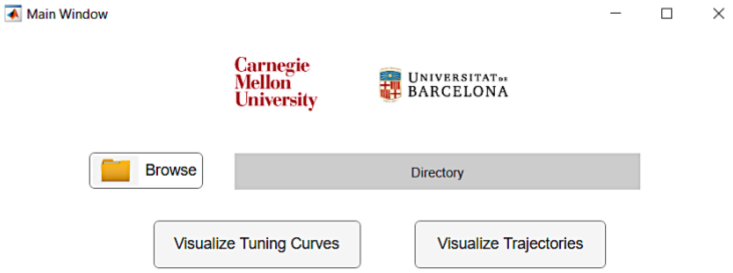
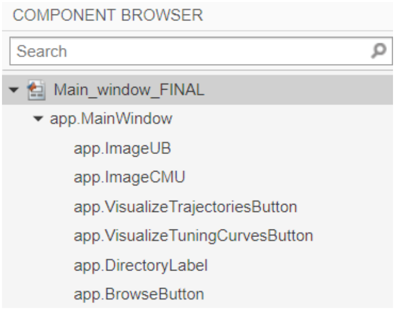


Figure 10. Main window

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.



COMPONENT BROWSER

Search

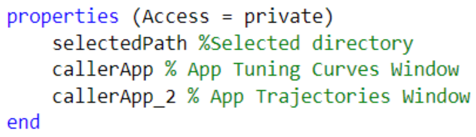
- Main_window_FINAL
  - app.MainWindow
    - app.ImageUB
    - app.ImageCMU
    - app.VisualizeTrajectoriesButton
    - app.VisualizeTuningCurvesButton
    - app.DirectoryLabel
    - app.BrowseButton

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.

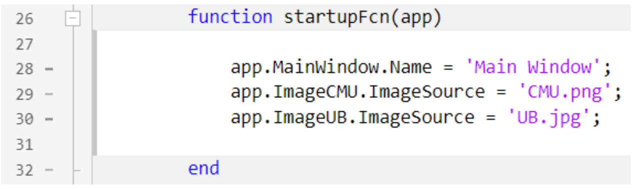


```
properties (Access = private)
    selectedPath %Selected directory
    callerApp % App Tuning Curves Window
    callerApp_2 % App Trajectories Window
end
```

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.



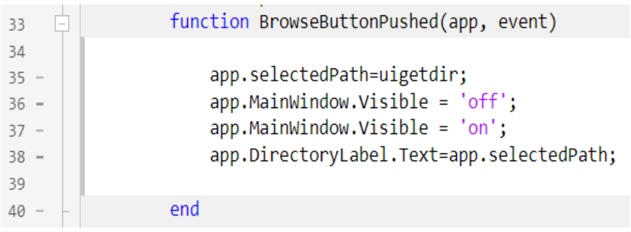
```
function startupFcn(app)

    app.MainWindow.Name = 'Main Window';
    app.ImageCMU.ImageSource = 'CMU.png';
    app.ImageUB.ImageSource = 'UB.jpg';

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.

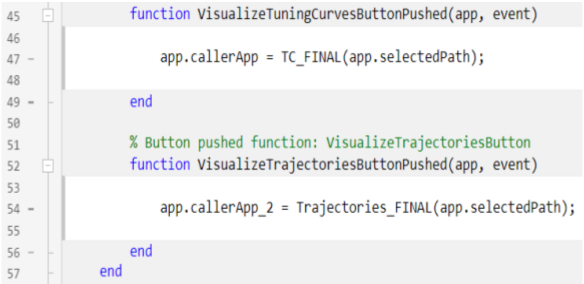


```
        function BrowseButtonPushed(app, event)

            app.selectedPath=uigetdir;
            app.MainWindow.Visible = 'off';
            app.MainWindow.Visible = 'on';
            app.DirectoryLabel.Text=app.selectedPath;

        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.



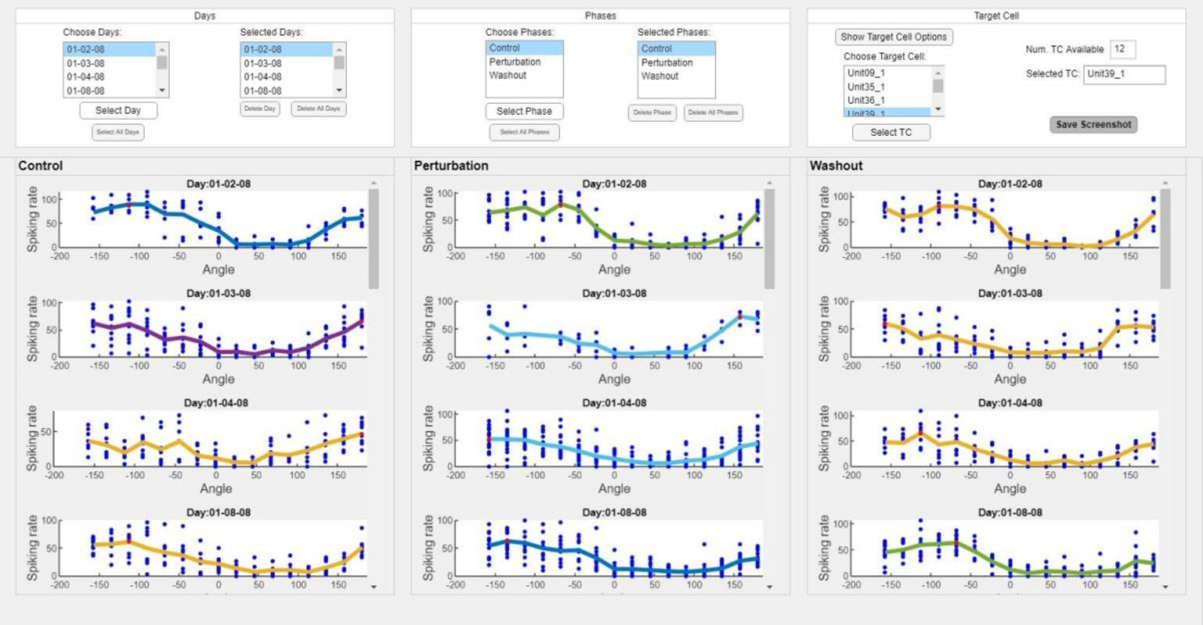
```
45          function VisualizeTuningCurvesButtonPushed(app, event)
46
47                  app.callerApp = TC_FINAL(app.selectedPath);
48
49          end
50
51          % Button pushed function: VisualizeTrajectoriesButton
52          function VisualizeTrajectoriesButtonPushed(app, event)
53
54  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



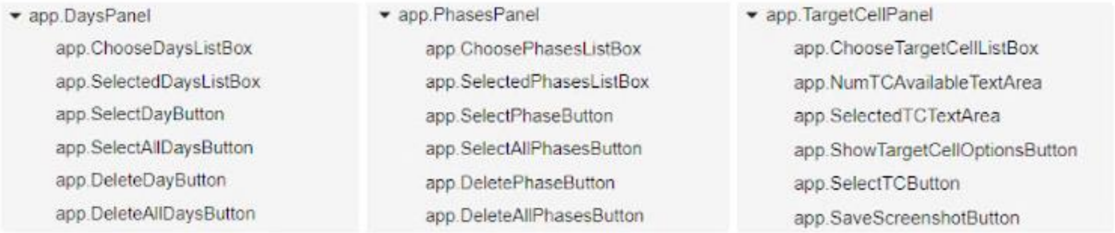
Num. TC Available: 12

Selected TC: Unit39_1

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.



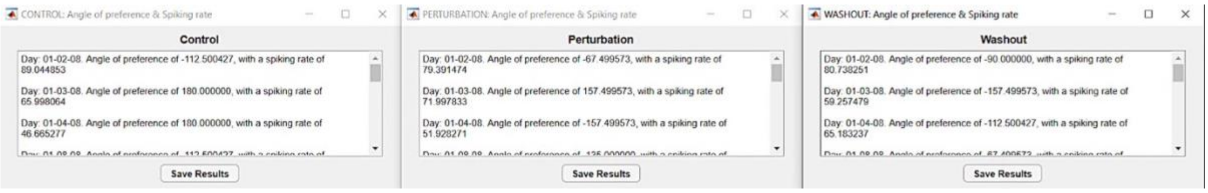
▾ app.DaysPanel
- app.ChooseDaysListBox
- app.SelectedDaysListBox
- app.SelectDayButton
- app.SelectAllDaysButton
- app.DeleteDayButton
- app.DeleteAllDaysButton

▾ app.PhasesPanel
- app.ChoosePhasesListBox
- app.SelectedPhasesListBox
- app.SelectPhaseButton
- app.SelectAllPhasesButton
- app.DeletePhaseButton
- app.DeleteAllPhasesButton

▾ app.TargetCellPanel
- app.ChooseTargetCellListBox
- app.NumTCAvailableTextArea
- app.SelectedTCTextArea
- app.ShowTargetCellOptionsButton
- app.SelectTCButton
- app.SaveScreenshotButton

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.



CONTROL: Angle of preference & Spiking rate

Control

Day: 01-02-08. Angle of preference of -112.500427, with a spiking rate of 89.044853

Day: 01-03-08. Angle of preference of 180.000000, with a spiking rate of 65.998064

Day: 01-04-08. Angle of preference of 180.000000, with a spiking rate of 46.665277

Day: 01-08-08. Angle of preference of -112.500427, with a spiking rate of

PERTURBATION: Angle of preference & Spiking rate

Perturbation

Day: 01-02-08. Angle of preference of -67.499573, with a spiking rate of 79.391474

Day: 01-03-08. Angle of preference of 157.499573, with a spiking rate of 71.997833

Day: 01-04-08. Angle of preference of -157.499573, with a spiking rate of 51.928271

Day: 01-08-08. Angle of preference of -135.000000, with a spiking rate of

WASHOUT: Angle of preference & Spiking rate

Washout

Day: 01-02-08. Angle of preference of -90.000000, with a spiking rate of 80.738251

Day: 01-03-08. Angle of preference of -157.499573, with a spiking rate of 59.257479

Day: 01-04-08. Angle of preference of -112.500427, with a spiking rate of 65.183237

Day: 01-08-08. Angle of preference of -67.499573, with a spiking rate of

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.

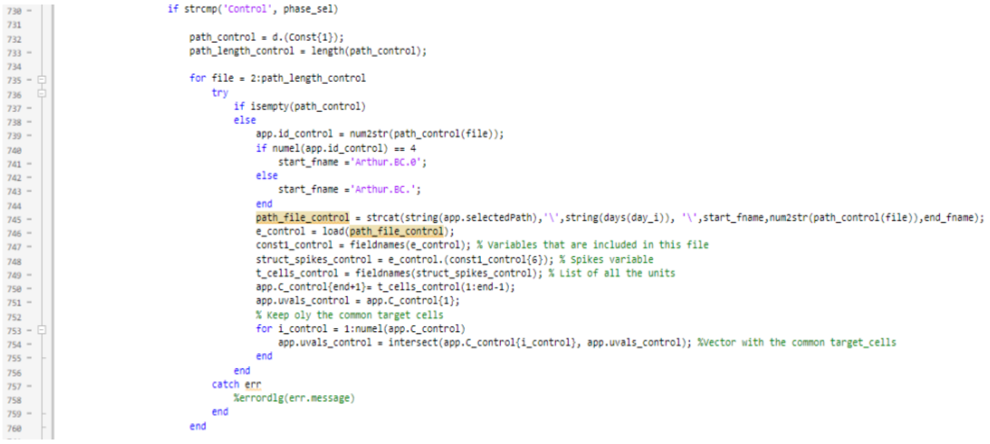
Callback functions

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.

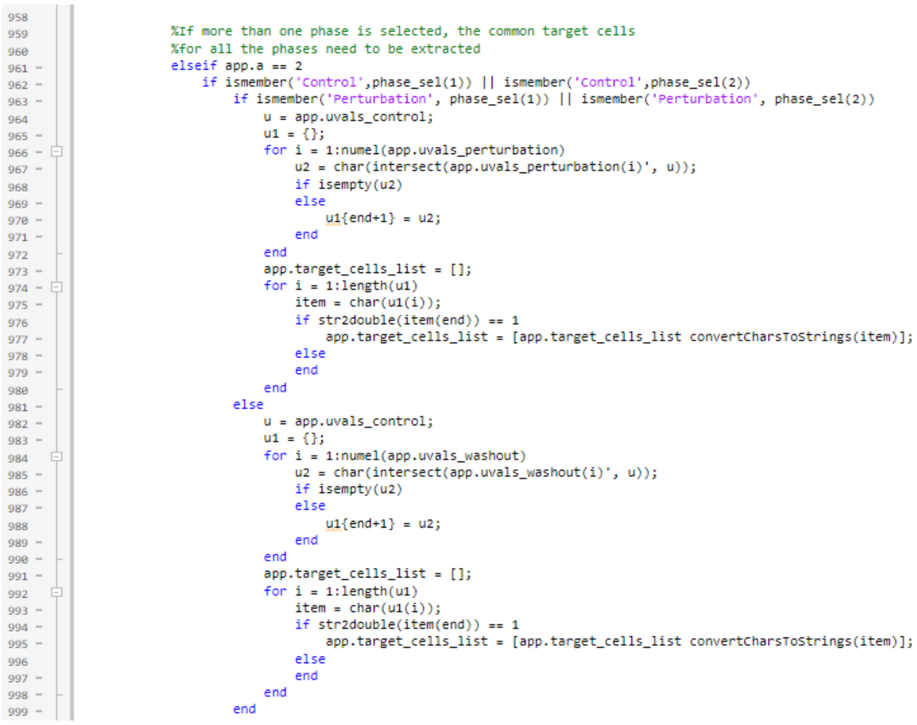


```
if strcmp('Control', phase_sel)

    path_control = d.(Const{1});
    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_i)), '\',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{1};
                % Keep oly 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
```

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.



```
%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 convertCharsToStrings(item)];
                else
                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 convertCharsToStrings(item)];
                else
                end
            end
        end
```

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

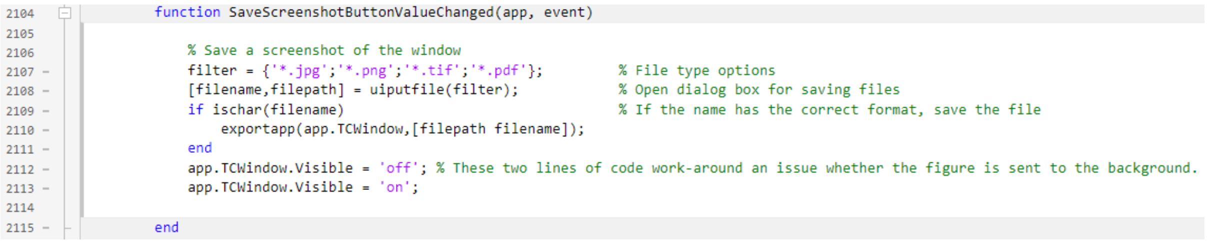
SelectTCButtonPushed(app, event)

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.



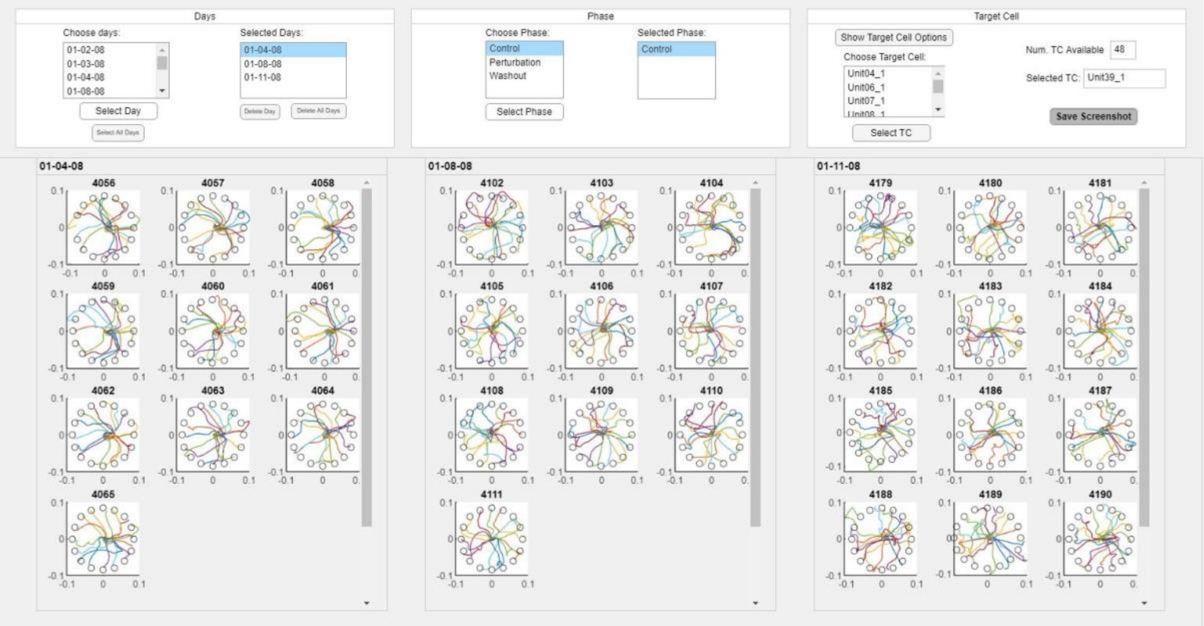
```
        function SaveScreenshotButtonValueChanged(app, event)

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

        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.



Days

Choose days: 01-02-08, 01-03-08, 01-04-08, 01-08-08

Selected Days: 01-04-08, 01-08-08, 01-11-08

Phase

Choose Phase: Control, Perturbation, Washout

Selected Phase: Control

Target Cell

Choose Target Cell: Unit04_1, Unit06_1, Unit07_1, Unit08_1

Num. TC Available 48

Selected TC: Unit39_1

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.

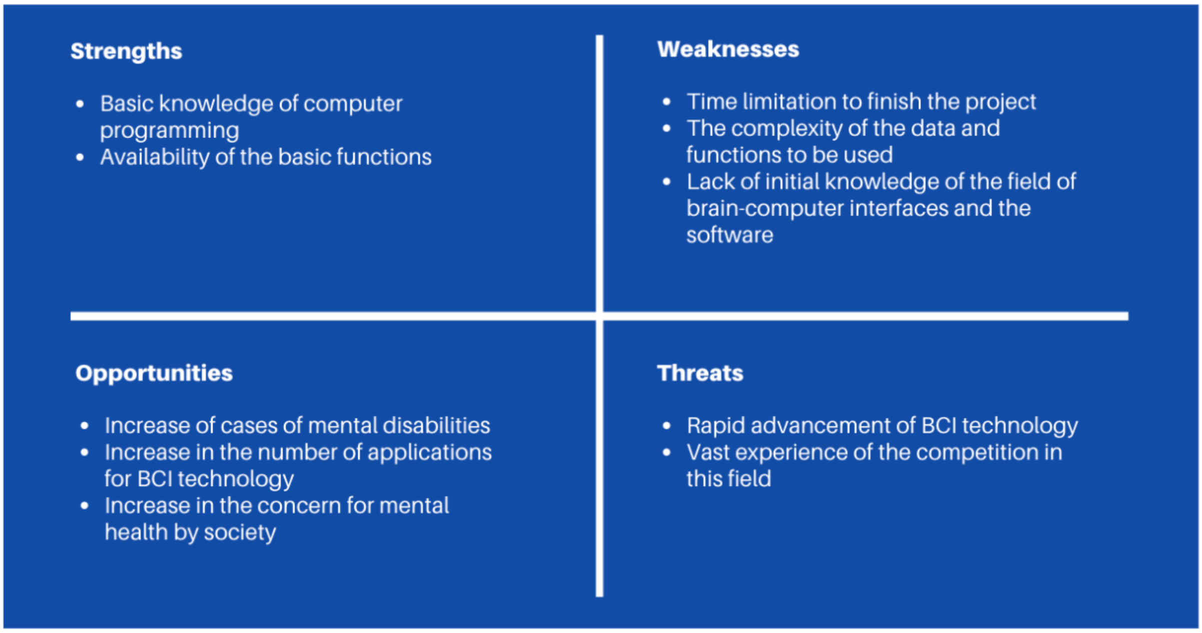
On the other hand, as the software generated is implemented as a set of *.mlapp* files (one for each generated window), another option to share the application would be to compress these files in a zip folder. To read such files, the end user will need to have the MATLAB license fully installed. It should 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 • Basic knowledge of computer programming • Availability of the basic functions	**Weaknesses** • 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 • 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** • 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.

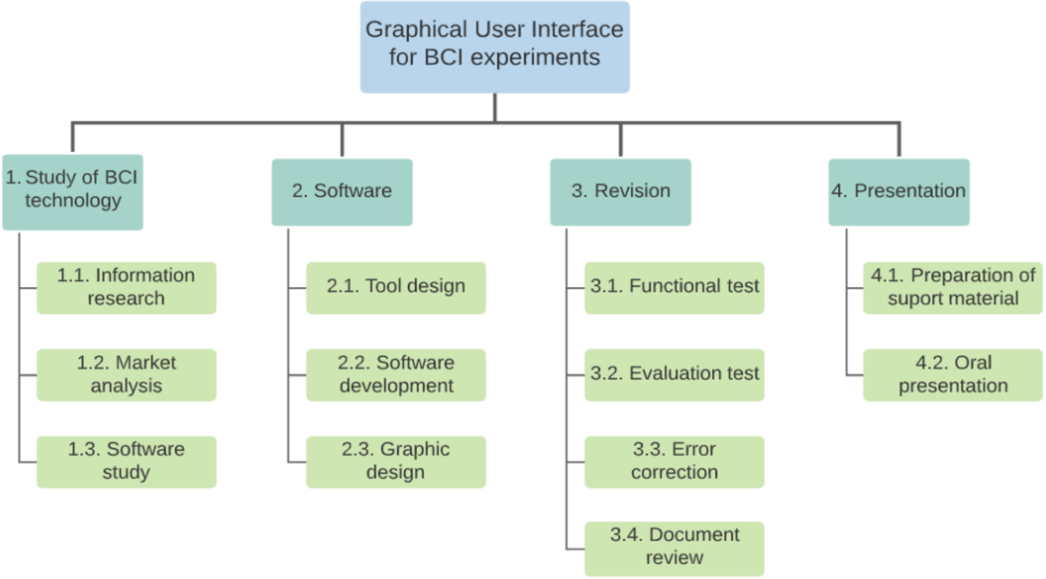
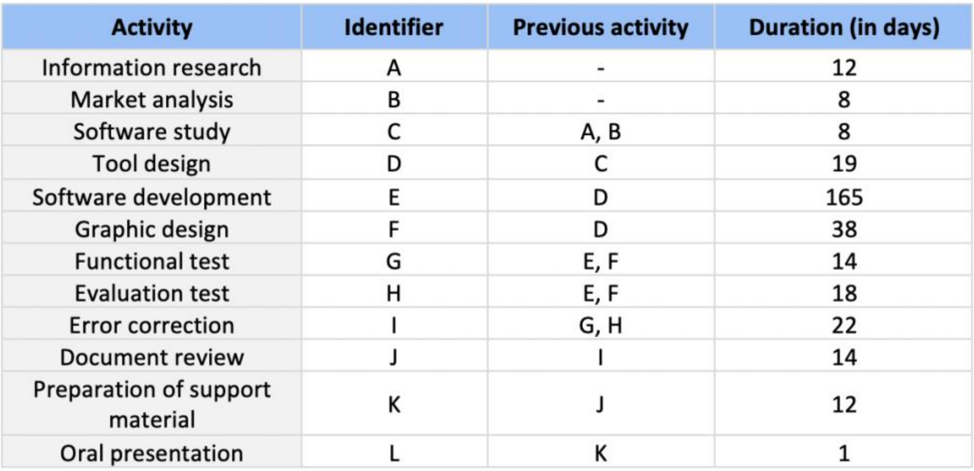


Figure 23. Work Breakdown Structure

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.

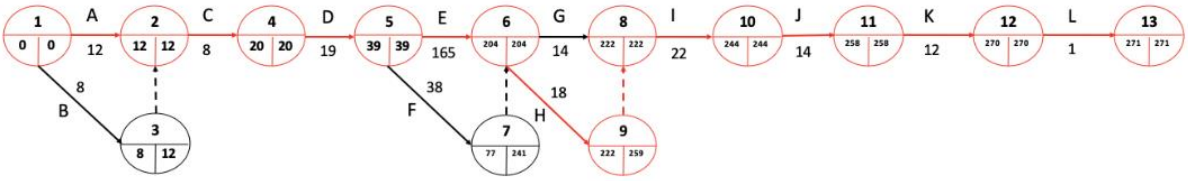


Figure 24. PERT diagram

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*

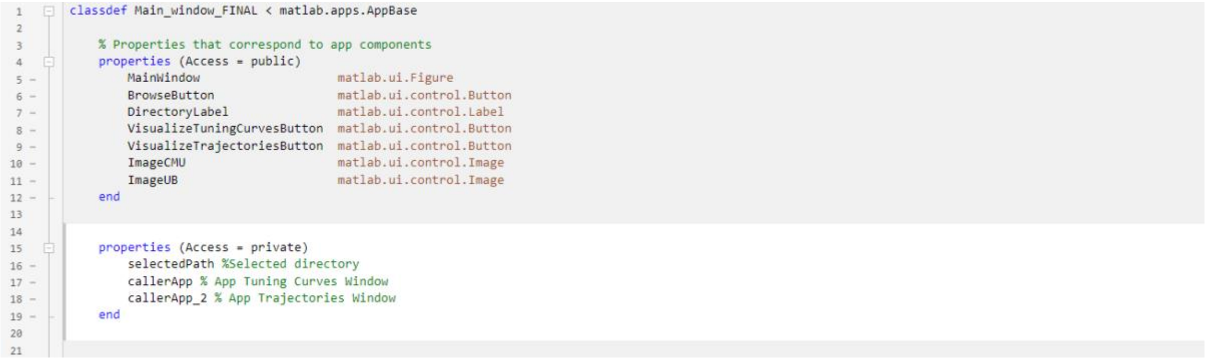
Transactions on Biomedical Engineering, *51*(6), 1034– 1043. https://doi.org/10.1109/TBME.2004.827072

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.

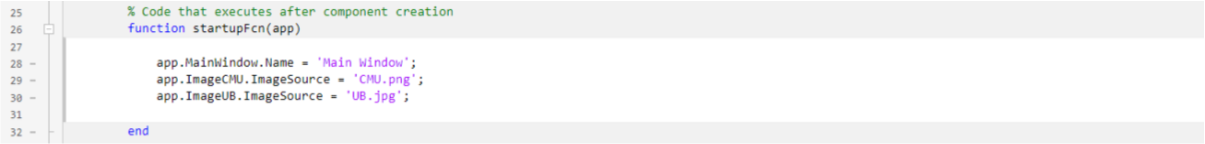
ANNEX I. Code used for the development of the main window

Poperties



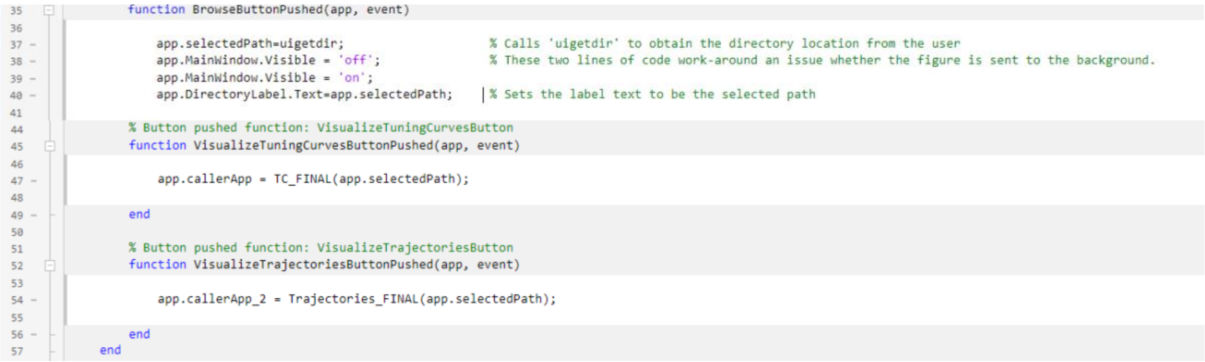
```
classdef Main_window_FINAL < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        MainWindow                   matlab.ui.Figure
        BrowseButton                 matlab.ui.control.Button
        DirectoryLabel               matlab.ui.control.Label
        VisualizeTuningCurvesButton  matlab.ui.control.Button
        VisualizeTrajectoriesButton  matlab.ui.control.Button
        ImageCMU                     matlab.ui.control.Image
        ImageUB                      matlab.ui.control.Image
    end


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

```

Startup function



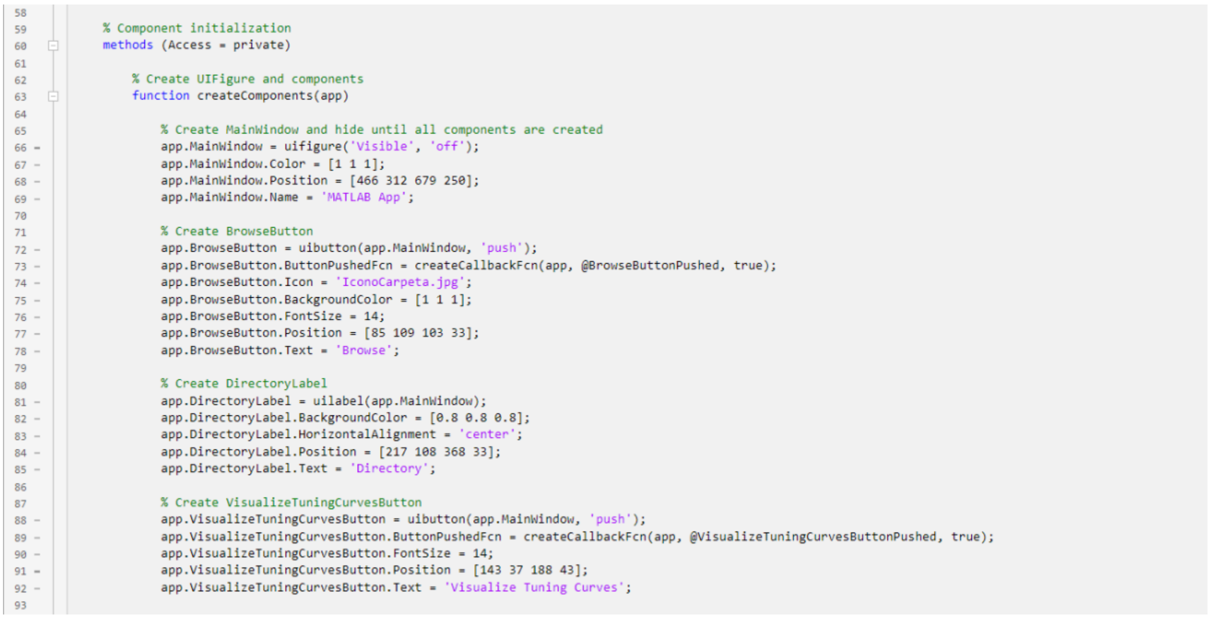
```
        % Code that executes after component creation
        function startupFcn(app)

            app.MainWindow.Name = 'Main Window';
            app.ImageCMU.ImageSource = 'CMU.png';
            app.ImageUB.ImageSource = 'UB.jpg';

        end
```

Callback functions

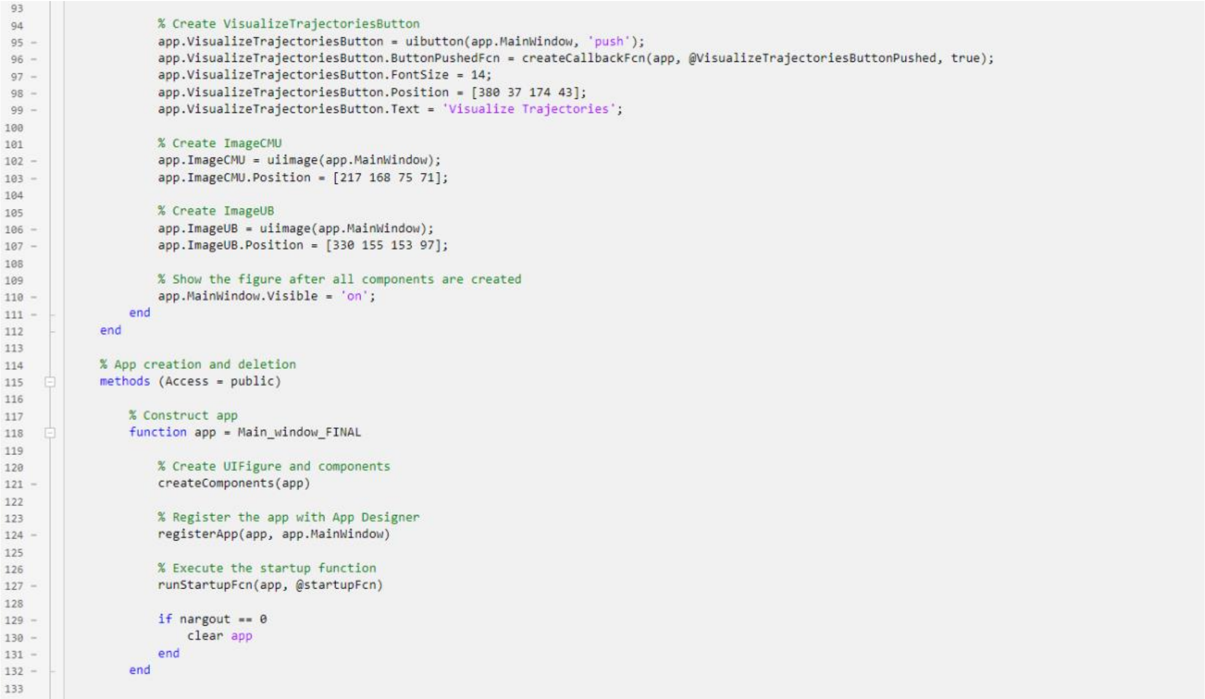


```
        function BrowseButtonPushed(app, event)

            app.selectedPath=uigetdir;                 % Calls 'uigetdir' to obtain the directory location from the user
            app.MainWindow.Visible = 'off';            % These two lines of code work-around an issue whether the figure is sent to the background.
            app.MainWindow.Visible = 'on';
            app.DirectoryLabel.Text=app.selectedPath;  % Sets the label text to be the selected path

        % Button pushed function: VisualizeTuningCurvesButton
        function VisualizeTuningCurvesButtonPushed(app, event)

            app.callerApp = TC_FINAL(app.selectedPath);

        end

        % Button pushed function: VisualizeTrajectoriesButton
        function VisualizeTrajectoriesButtonPushed(app, event)

            app.callerApp_2 = Trajectories_FINAL(app.selectedPath);

        end
    end
```

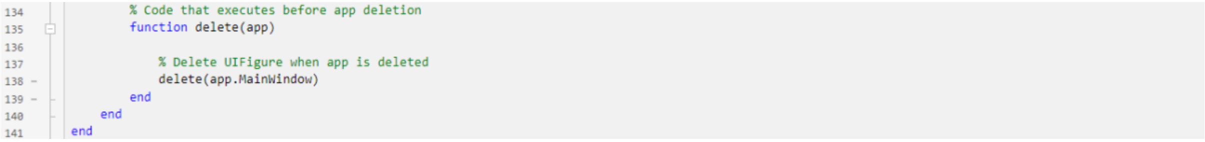
Component initialization



```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create MainWindow and hide until all components are created
            app.MainWindow = uifigure('Visible', 'off');
            app.MainWindow.Color = [1 1 1];
            app.MainWindow.Position = [466 312 679 250];
            app.MainWindow.Name = 'MATLAB App';

            % Create BrowseButton
            app.BrowseButton = uibutton(app.MainWindow, 'push');
            app.BrowseButton.ButtonPushedFcn = createCallbackFcn(app, @BrowseButtonPushed, true);
            app.BrowseButton.Icon = 'IconoCarpeta.jpg';
            app.BrowseButton.BackgroundColor = [1 1 1];
            app.BrowseButton.FontSize = 14;
            app.BrowseButton.Position = [85 109 103 33];
            app.BrowseButton.Text = 'Browse';

            % Create DirectoryLabel
            app.DirectoryLabel = uilabel(app.MainWindow);
            app.DirectoryLabel.BackgroundColor = [0.8 0.8 0.8];
            app.DirectoryLabel.HorizontalAlignment = 'center';
            app.DirectoryLabel.Position = [217 108 368 33];
            app.DirectoryLabel.Text = 'Directory';

            % Create VisualizeTuningCurvesButton
            app.VisualizeTuningCurvesButton = uibutton(app.MainWindow, 'push');
            app.VisualizeTuningCurvesButton.ButtonPushedFcn = createCallbackFcn(app, @VisualizeTuningCurvesButtonPushed, true);
            app.VisualizeTuningCurvesButton.FontSize = 14;
            app.VisualizeTuningCurvesButton.Position = [143 37 188 43];
            app.VisualizeTuningCurvesButton.Text = 'Visualize Tuning Curves';

```



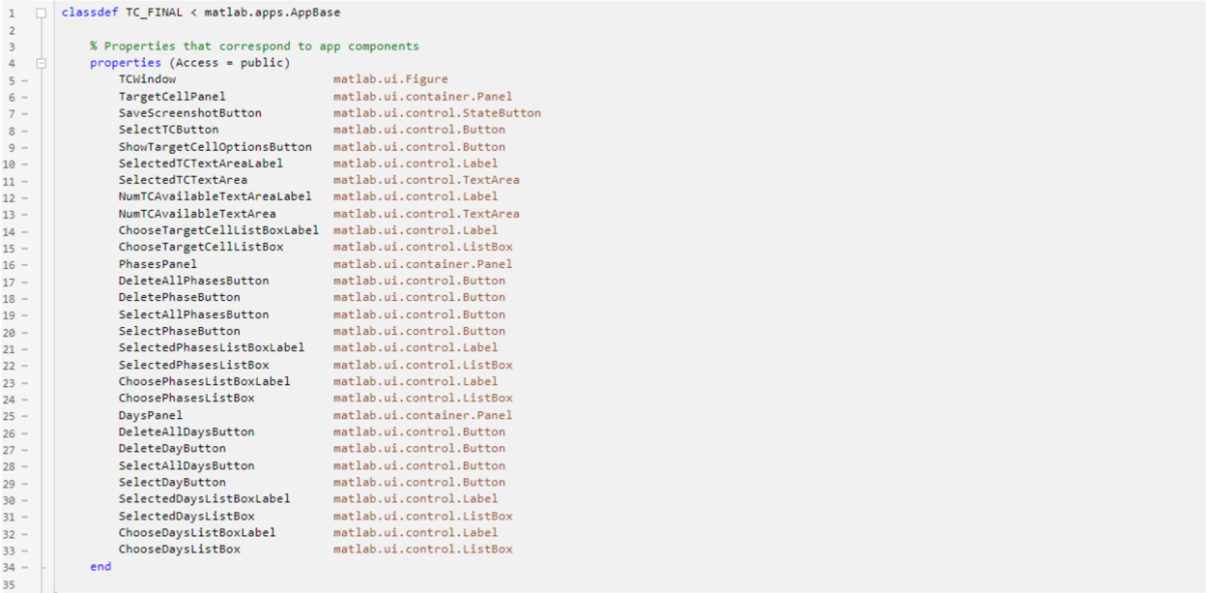
```

            % Create VisualizeTrajectoriesButton
            app.VisualizeTrajectoriesButton = uibutton(app.MainWindow, 'push');
            app.VisualizeTrajectoriesButton.ButtonPushedFcn = createCallbackFcn(app, @VisualizeTrajectoriesButtonPushed, true);
            app.VisualizeTrajectoriesButton.FontSize = 14;
            app.VisualizeTrajectoriesButton.Position = [380 37 174 43];
            app.VisualizeTrajectoriesButton.Text = 'Visualize Trajectories';

            % Create ImageCMU
            app.ImageCMU = uiimage(app.MainWindow);
            app.ImageCMU.Position = [217 168 75 71];

            % Create ImageUB
            app.ImageUB = uiimage(app.MainWindow);
            app.ImageUB.Position = [330 155 153 97];

            % Show the figure after all components are created
            app.MainWindow.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Main_window_FINAL

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.MainWindow)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

```



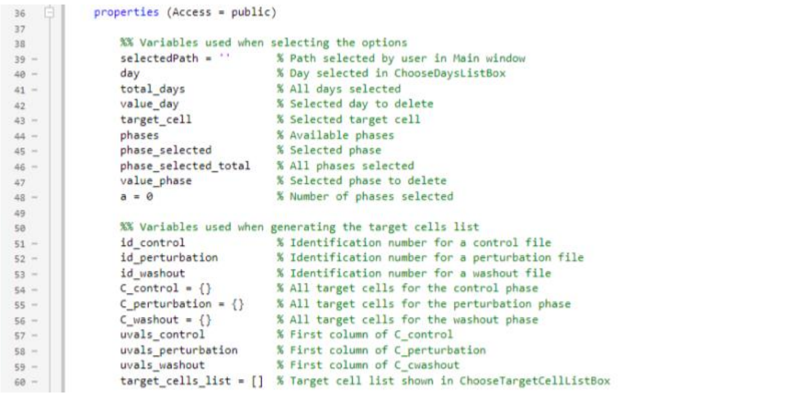
```
        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.MainWindow)
        end
    end
end
```

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

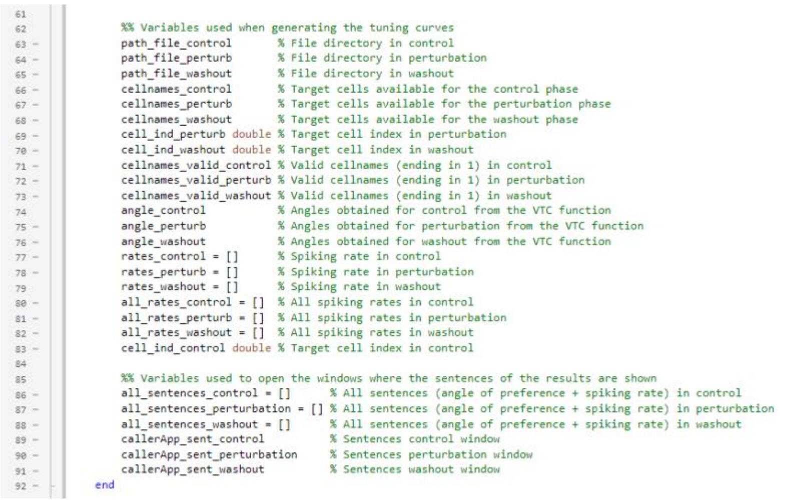
Properties



```
classdef TC_FINAL < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        TCWindow                        matlab.ui.Figure
        TargetCellPanel                 matlab.ui.container.Panel
        SaveScreenshotButton            matlab.ui.control.StateButton
        SelectTCButton                  matlab.ui.control.Button
        ShowTargetCellOptionsButton     matlab.ui.control.Button
        SelectedTCTextAreaLabel         matlab.ui.control.Label
        SelectedTCTextArea              matlab.ui.control.TextArea
        NumTCAvailableTextAreaLabel     matlab.ui.control.Label
        NumTCAvailableTextArea          matlab.ui.control.TextArea
        ChooseTargetCellListBoxLabel    matlab.ui.control.Label
        ChooseTargetCellListBox         matlab.ui.control.ListBox
        PhasesPanel                     matlab.ui.container.Panel
        DeleteAllPhasesButton           matlab.ui.control.Button
        DeletePhaseButton               matlab.ui.control.Button
        SelectAllPhasesButton           matlab.ui.control.Button
        SelectPhaseButton               matlab.ui.control.Button
        SelectedPhasesListBoxLabel      matlab.ui.control.Label
        SelectedPhasesListBox           matlab.ui.control.ListBox
        ChoosePhasesListBoxLabel        matlab.ui.control.Label
        ChoosePhasesListBox             matlab.ui.control.ListBox
        DaysPanel                       matlab.ui.container.Panel
        DeleteAllDaysButton             matlab.ui.control.Button
        DeleteDayButton                 matlab.ui.control.Button
        SelectAllDaysButton             matlab.ui.control.Button
        SelectDayButton                 matlab.ui.control.Button
        SelectedDaysListBoxLabel        matlab.ui.control.Label
        SelectedDaysListBox             matlab.ui.control.ListBox
        ChooseDaysListBoxLabel          matlab.ui.control.Label
        ChooseDaysListBox               matlab.ui.control.ListBox
    end

```

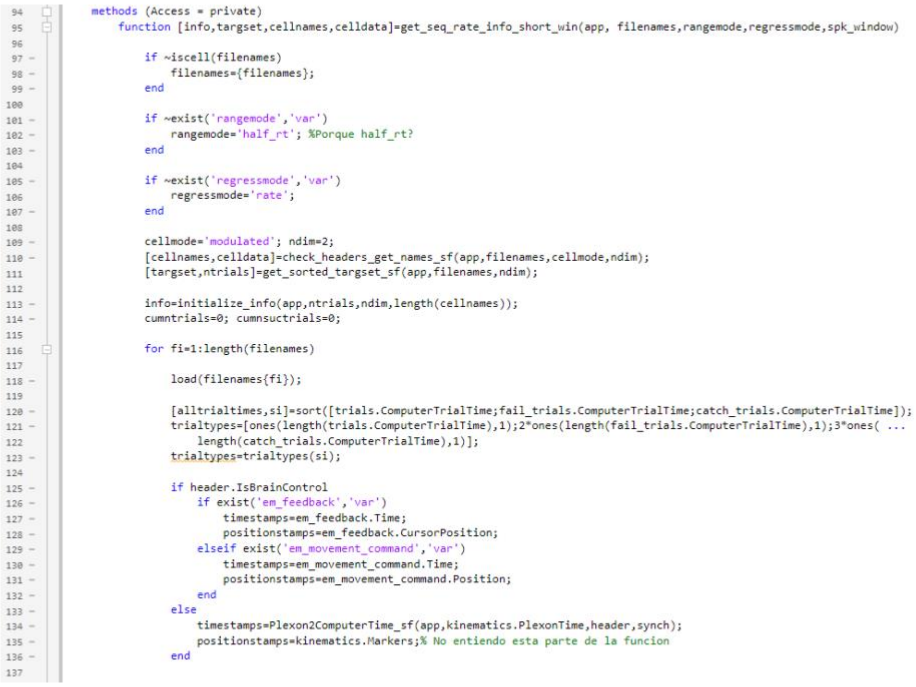


```
    properties (Access = public)

        %% Variables used when selecting the options
        selectedPath = ''       % Path selected by user in Main window
        day                     % Day selected in ChooseDaysListBox
        total_days              % All days selected
        value_day               % Selected day to delete
        target_cell             % Selected target cell
        phases                  % Available phases
        phase_selected          % Selected phase
        phase_selected_total    % All phases selected
        value_phase             % Selected phase to delete
        a = 0                   % Number of phases selected

        %% Variables used when generating the target cells list
        id_control              % Identification number for a control file
        id_perturbation         % Identification number for a perturbation file
        id_washout              % Identification number for a washout file
        C_control = {}          % All target cells for the control phase
        C_perturbation = {}     % All target cells for the perturbation phase
        C_washout = {}          % All target cells for the washout phase
        uvals_control           % First column of C_control
        uvals_perturbation      % First column of C_perturbation
        uvals_washout           % First column of C_cwashout
        target_cells_list = []  % Target cell list shown in ChooseTargetCellListBox
```

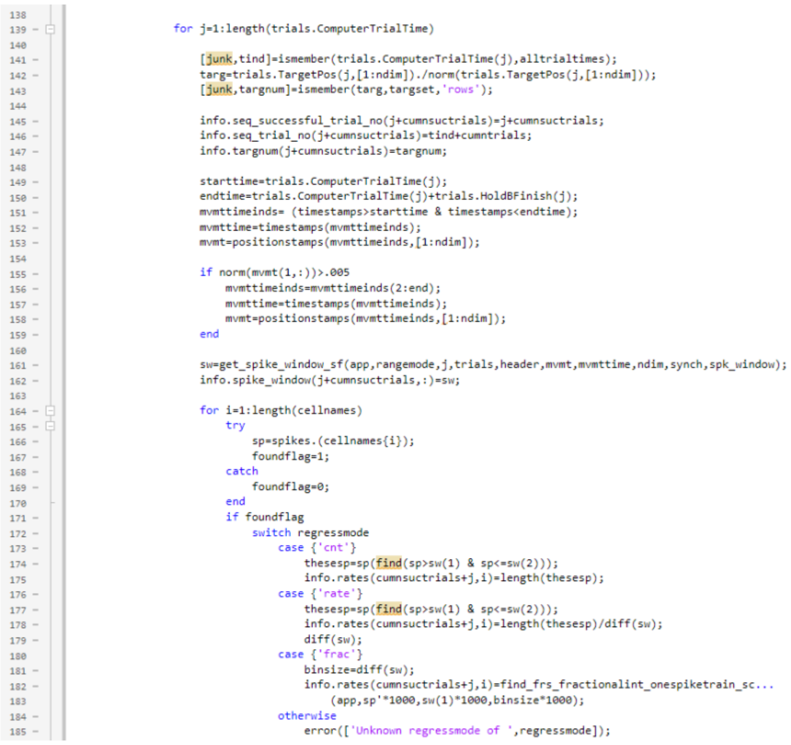


```

        %% Variables used when generating the tuning curves
        path_file_control       % File directory in control
        path_file_perturb       % File directory in perturbation
        path_file_washout       % File directory in washout
        cellnames_control       % Target cells available for the control phase
        cellnames_perturb       % Target cells available for the perturbation phase
        cellnames_washout       % Target cells available for the washout phase
        cell_ind_perturb double % Target cell index in perturbation
        cell_ind_washout double % Target cell index in washout
        cellnames_valid_control % Valid cellnames (ending in 1) in control
        cellnames_valid_perturb % Valid cellnames (ending in 1) in perturbation
        cellnames_valid_washout % Valid cellnames (ending in 1) in washout
        angle_control           % Angles obtained for control from the VTC function
        angle_perturb           % Angles obtained for perturbation from the VTC function
        angle_washout           % Angles obtained for washout from the VTC function
        rates_control = []      % Spiking rate in control
        rates_perturb = []      % Spiking rate in perturbation
        rates_washout = []      % Spiking rate in washout
        all_rates_control = []  % All spiking rates in control
        all_rates_perturb = []  % All spiking rates in perturbation
        all_rates_washout = []  % All spiking rates in washout
        cell_ind_control double % Target cell index in control

        %% Variables used to open the windows where the sentences of the results are shown
        all_sentences_control = []      % All sentences (angle of preference + spiking rate) in control
        all_sentences_perturbation = [] % All sentences (angle of preference + spiking rate) in perturbation
        all_sentences_washout = []      % All sentences (angle of preference + spiking rate) in washout
        callerApp_sent_control          % Sentences control window
        callerApp_sent_perturbation     % Sentences perturbation window
        callerApp_sent_washout          % Sentences washout window
    end
```

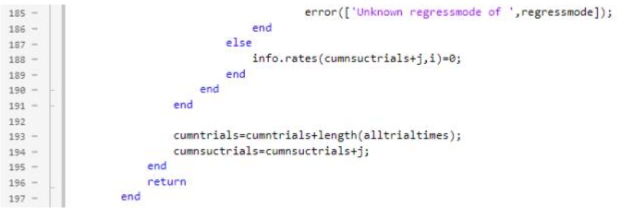
Methods



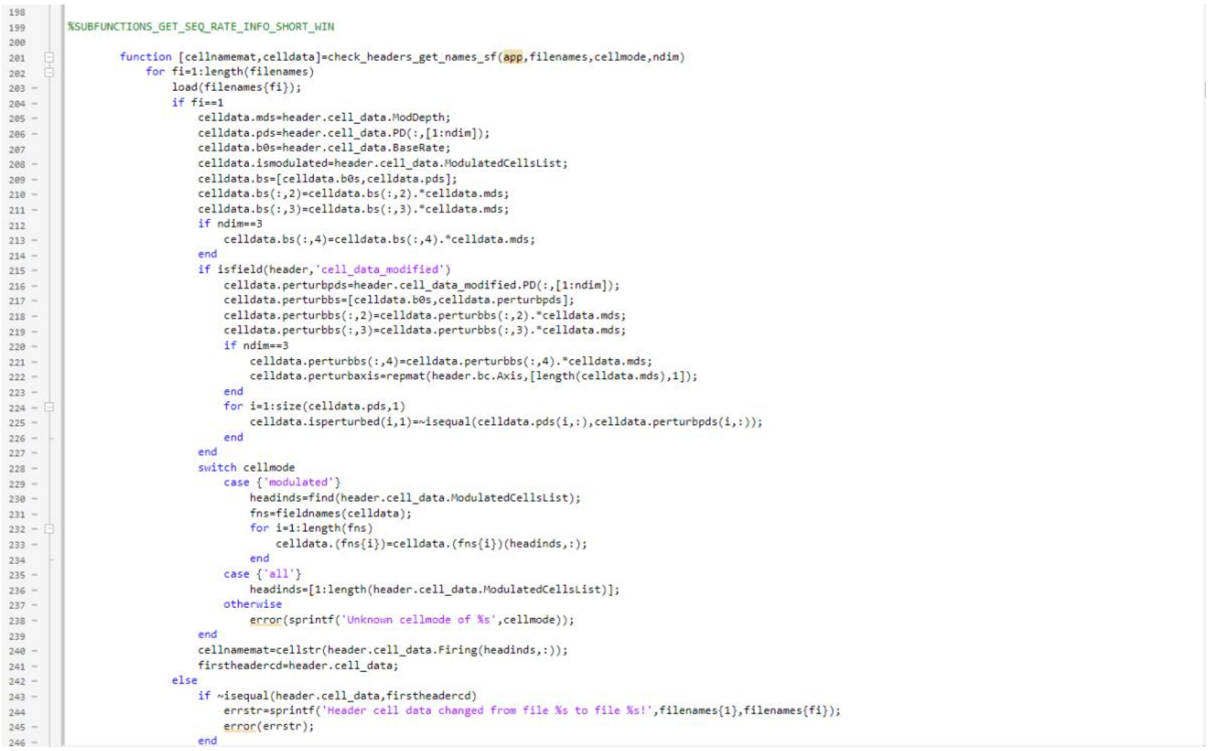
```
methods (Access = private)
    function [info,targset,cellnames,celldata]=get_seq_rate_info_short_win(app, filenames,rangemode,regressmode,spk_window)

        if ~iscell(filenames)
            filenames={filenames};
        end

        if ~exist('rangemode','var')
            rangemode='half_rt'; %Porque half_rt?
        end

        if ~exist('regressmode','var')
            regressmode='rate';
        end

        cellmode='modulated'; ndim=2;
        [cellnames,celldata]=check_headers_get_names_sf(app,filenames,cellmode,ndim);
        [targset,ntrials]=get_sorted_targset_sf(app,filenames,ndim);

        info=initialize_info(app,ntrials,ndim,length(cellnames));
        cumntrials=0; cumnsuctrials=0;

        for fi=1:length(filenames)

            load(filenames{fi});

            [alltrialtimes,si]=sort([trials.ComputerTrialTime;fail_trials.ComputerTrialTime;catch_trials.ComputerTrialTime]);
            trialtypes=[ones(length(trials.ComputerTrialTime),1);2*ones(length(fail_trials.ComputerTrialTime),1);3*ones( ...
                length(catch_trials.ComputerTrialTime),1)];
            trialtypes=trialtypes(si);

            if header.IsBrainControl
                if exist('em_feedback','var')
                    timestamps=em_feedback.Time;
                    positionstamps=em_feedback.CursorPosition;
                elseif exist('em_movement_command','var')
                    timestamps=em_movement_command.Time;
                    positionstamps=em_movement_command.Position;
                end
            else
                timestamps=Plexon2ComputerTime_sf(app,kinematics.PlexonTime,header,synch);
                positionstamps=kinematics.Markers;% No entiendo esta parte de la funcion
            end

```



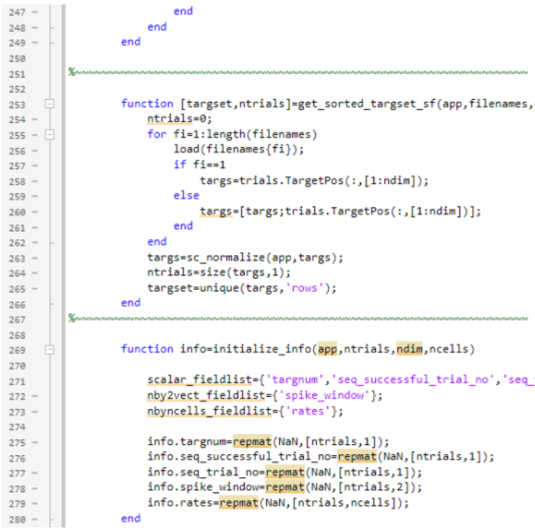
```

            for j=1:length(trials.ComputerTrialTime)

                [junk,tind]=ismember(trials.ComputerTrialTime(j),alltrialtimes);
                targ=trials.TargetPos(j,[1:ndim])./norm(trials.TargetPos(j,[1:ndim]));
                [junk,targnum]=ismember(targ,targset,'rows');

                info.seq_successful_trial_no(j+cumnsuctrials)=j+cumnsuctrials;
                info.seq_trial_no(j+cumnsuctrials)=tind+cumntrials;
                info.targnum(j+cumnsuctrials)=targnum;

                starttime=trials.ComputerTrialTime(j);
                endtime=trials.ComputerTrialTime(j)+trials.HoldBFinish(j);
                mvmttimeinds= (timestamps>starttime & timestamps<endtime);
                mvmttime=timestamps(mvmttimeinds);
                mvmt=positionstamps(mvmttimeinds,[1:ndim]);

                if norm(mvmt(1,:))>.005
                    mvmttimeinds=mvmttimeinds(2:end);
                    mvmttime=timestamps(mvmttimeinds);
                    mvmt=positionstamps(mvmttimeinds,[1:ndim]);
                end

                sw=get_spike_window_sf(app,rangemode,j,trials,header,mvmt,mvmttime,ndim,synch,spk_window);
                info.spike_window(j+cumnsuctrials,:)=sw;

                for i=1:length(cellnames)
                    try
                        sp=spikes.(cellnames{i});
                        foundflag=1;
                    catch
                        foundflag=0;
                    end
                    if foundflag
                        switch regressmode
                            case {'cnt'}
                                thesesp=sp(find(sp>sw(1) & sp<=sw(2)));
                                info.rates(cumnsuctrials+j,i)=length(thesesp);
                            case {'rate'}
                                thesesp=sp(find(sp>sw(1) & sp<=sw(2)));
                                info.rates(cumnsuctrials+j,i)=length(thesesp)/diff(sw);
                                diff(sw);
                            case {'frac'}
                                binsize=diff(sw);
                                info.rates(cumnsuctrials+j,i)=find_frs_fractionalint_onespiketrain_sc...
                                    (app,sp'*1000,sw(1)*1000,binsize*1000);
                            otherwise
                                error(['Unknown regressmode of ',regressmode]);
```



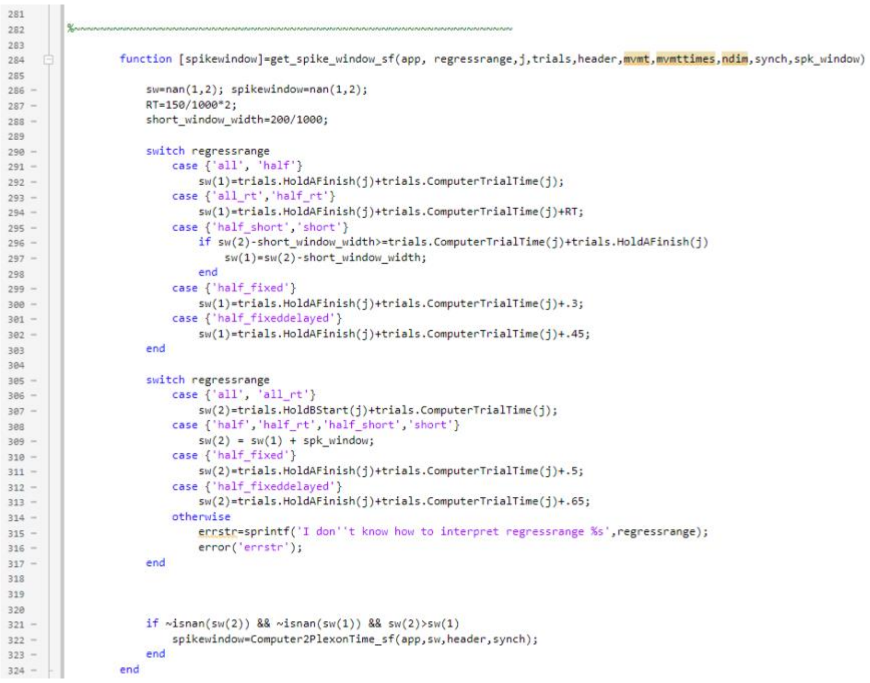
```
                                error(['Unknown regressmode of ',regressmode]);
                        end
                    else
                        info.rates(cumnsuctrials+j,i)=0;
                    end
                end
            end

            cumntrials=cumntrials+length(alltrialtimes);
            cumnsuctrials=cumnsuctrials+j;
        end
        return
    end
```



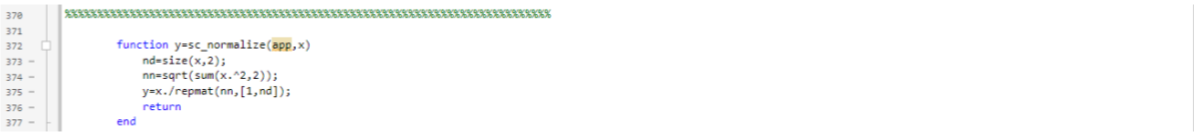
```
%SUBFUNCTIONS_GET_SEQ_RATE_INFO_SHORT_WIN

        function [cellnamemat,celldata]=check_headers_get_names_sf(app,filenames,cellmode,ndim)
            for fi=1:length(filenames)
                load(filenames{fi});
                if fi==1
                    celldata.mds=header.cell_data.ModDepth;
                    celldata.pds=header.cell_data.PD(:,[1:ndim]);
                    celldata.b0s=header.cell_data.BaseRate;
                    celldata.ismodulated=header.cell_data.ModulatedCellsList;
                    celldata.bs=[celldata.b0s,celldata.pds];
                    celldata.bs(:,2)=celldata.bs(:,2).*celldata.mds;
                    celldata.bs(:,3)=celldata.bs(:,3).*celldata.mds;
                    if ndim==3
                        celldata.bs(:,4)=celldata.bs(:,4).*celldata.mds;
                    end
                    if isfield(header,'cell_data_modified')
                        celldata.perturbpds=header.cell_data_modified.PD(:,[1:ndim]);
                        celldata.perturbbs=[celldata.b0s,celldata.perturbpds];
                        celldata.perturbbs(:,2)=celldata.perturbbs(:,2).*celldata.mds;
                        celldata.perturbbs(:,3)=celldata.perturbbs(:,3).*celldata.mds;
                        if ndim==3
                            celldata.perturbbs(:,4)=celldata.perturbbs(:,4).*celldata.mds;
                            celldata.perturbaxis=repmat(header.bc.Axis,[length(celldata.mds),1]);
                        end
                        for i=1:size(celldata.pds,1)
                            celldata.isperturbed(i,1)=~isequal(celldata.pds(i,:),celldata.perturbpds(i,:));
                        end
                    end
                    switch cellmode
                        case {'modulated'}
                            headinds=find(header.cell_data.ModulatedCellsList);
                            fns=fieldnames(celldata);
                            for i=1:length(fns)
                                celldata.(fns{i})=celldata.(fns{i})(headinds,:);
                            end
                        case {'all'}
                            headinds=[1:length(header.cell_data.ModulatedCellsList)];
                        otherwise
                            error(sprintf('Unknown cellmode of %s',cellmode));
                    end
                    cellnamemat=cellstr(header.cell_data.Firing(headinds,:));
                    firstheadercd=header.cell_data;
                else
                    if ~isequal(header.cell_data,firstheadercd)
                        errstr=sprintf('Header cell data changed from file %s to file %s!',filenames{1},filenames{fi});
                        error(errstr);
                    end
```



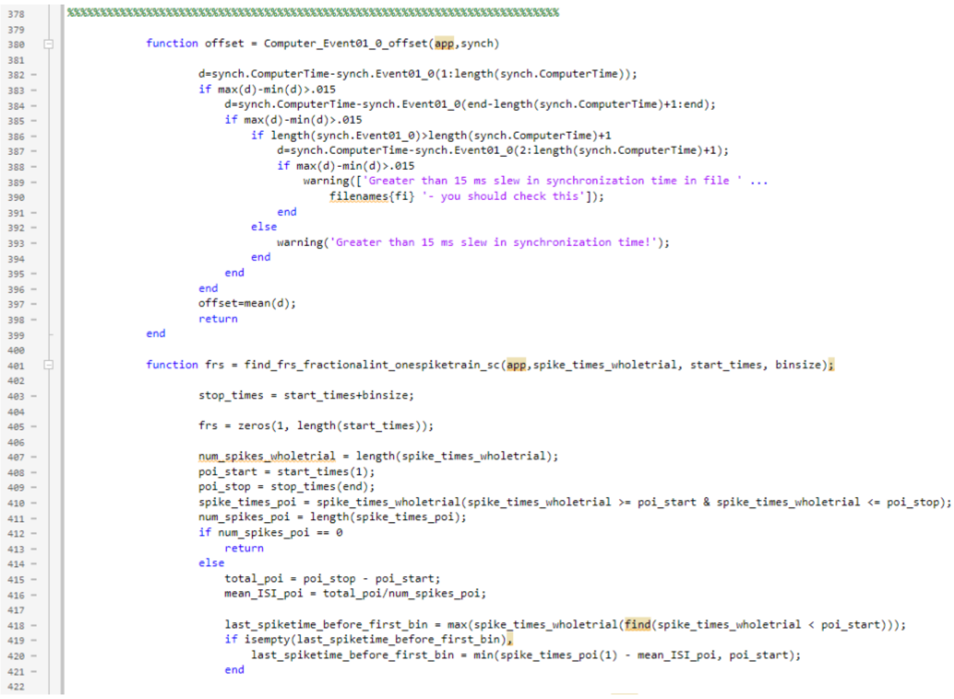
```
                end
            end
        end

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        function [targset,ntrials]=get_sorted_targset_sf(app,filenames,ndim)
            ntrials=0;
            for fi=1:length(filenames)
                load(filenames{fi});
                if fi==1
                    targs=trials.TargetPos(:,[1:ndim]);
                else
                    targs=[targs;trials.TargetPos(:,[1:ndim])];
                end
            end
            targs=sc_normalize(app,targs);
            ntrials=size(targs,1);
            targset=unique(targs,'rows');
        end
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        function info=initialize_info(app,ntrials,ndim,ncells)

            scalar_fieldlist={'targnum','seq_successful_trial_no','seq_trial_no'};
            nby2vect_fieldlist={'spike_window'};
            nbyncells_fieldlist={'rates'};

            info.targnum=repmat(NaN,[ntrials,1]);
            info.seq_successful_trial_no=repmat(NaN,[ntrials,1]);
            info.seq_trial_no=repmat(NaN,[ntrials,1]);
            info.spike_window=repmat(NaN,[ntrials,2]);
            info.rates=repmat(NaN,[ntrials,ncells]);
        end
```



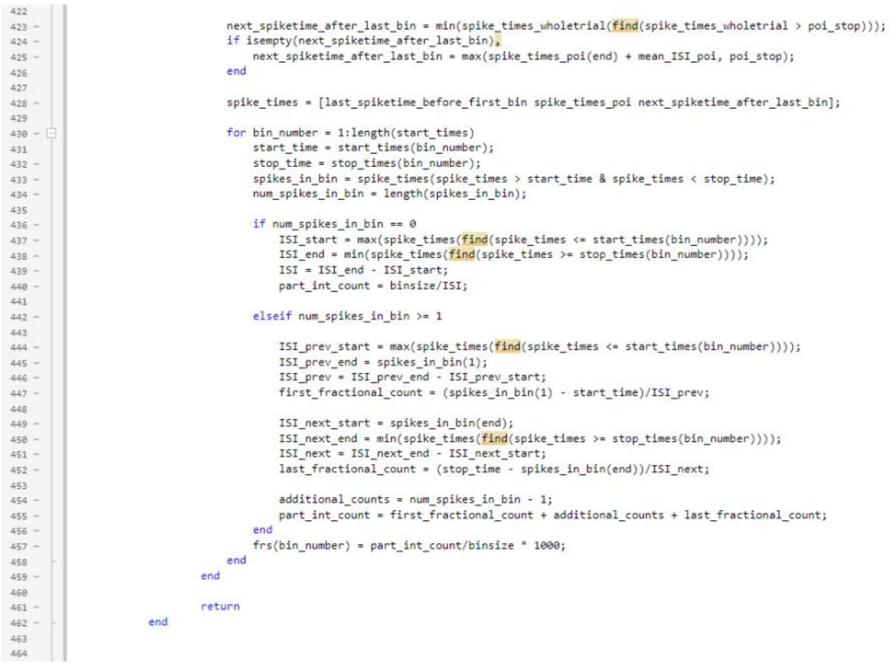
```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        function [spikewindow]=get_spike_window_sf(app, regressrange,j,trials,header,mvmt,mvmttimes,ndim,synch,spk_window)

            sw=nan(1,2); spikewindow=nan(1,2);
            RT=150/1000*2;
            short_window_width=200/1000;

            switch regressrange
                case {'all', 'half'}
                    sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j);
                case {'all_rt','half_rt'}
                    sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+RT;
                case {'half_short','short'}
                    if sw(2)-short_window_width>=trials.ComputerTrialTime(j)+trials.HoldAFinish(j)
                        sw(1)=sw(2)-short_window_width;
                    end
                case {'half_fixed'}
                    sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.3;
                case {'half_fixeddelayed'}
                    sw(1)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.45;
            end

            switch regressrange
                case {'all', 'all_rt'}
                    sw(2)=trials.HoldBStart(j)+trials.ComputerTrialTime(j);
                case {'half','half_rt','half_short','short'}
                    sw(2) = sw(1) + spk_window;
                case {'half_fixed'}
                    sw(2)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.5;
                case {'half_fixeddelayed'}
                    sw(2)=trials.HoldAFinish(j)+trials.ComputerTrialTime(j)+.65;
                otherwise
                    errstr=sprintf('I don''t know how to interpret regressrange %s',regressrange);
                    error('errstr');
            end



            if ~isnan(sw(2)) && ~isnan(sw(1)) && sw(2)>sw(1)
                spikewindow=Computer2PlexonTime_sf(app,sw,header,synch);
            end
        end
```



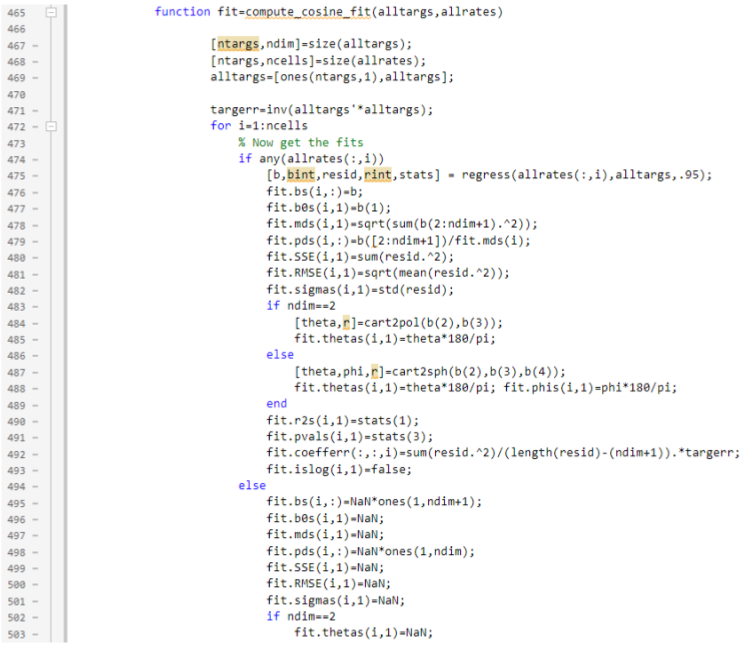
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        function y=sc_normalize(app,x)
            nd=size(x,2);
            nn=sqrt(sum(x.^2,2));
            y=x./repmat(nn,[1,nd]);
            return
        end
```



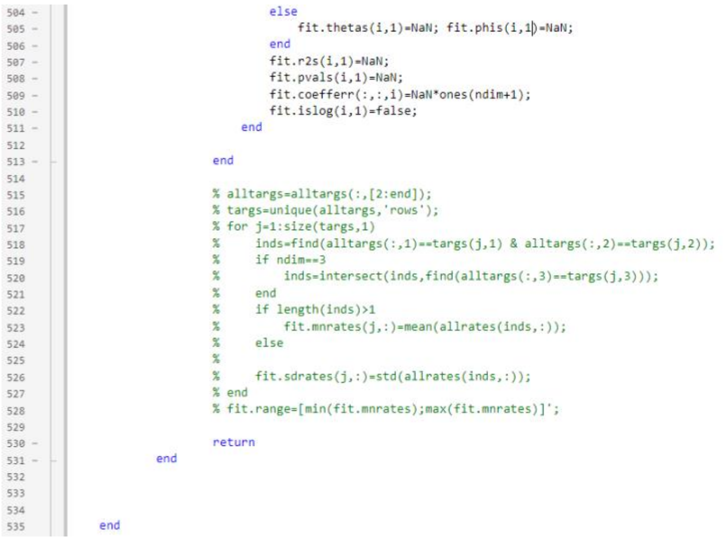
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            function offset = Computer_Event01_0_offset(app,synch)

                    d=synch.ComputerTime-synch.Event01_0(1:length(synch.ComputerTime));
                    if max(d)-min(d)>.015
                        d=synch.ComputerTime-synch.Event01_0(end-length(synch.ComputerTime)+1:end);
                        if max(d)-min(d)>.015
                            if length(synch.Event01_0)>length(synch.ComputerTime)+1
                                d=synch.ComputerTime-synch.Event01_0(2:length(synch.ComputerTime)+1);
                                if max(d)-min(d)>.015
                                    warning(['Greater than 15 ms slew in synchronization time in file ' ...
                                        filenames{fi} '- you should check this']);
                                end
                            else
                                warning('Greater than 15 ms slew in synchronization time!');
                            end
                        end
                    end
                    offset=mean(d);
                    return
            end

            function frs = find_frs_fractionalint_onespiketrain_sc(app,spike_times_wholetrial, start_times, binsize);

                    stop_times = start_times+binsize;

                    frs = zeros(1, length(start_times));

                    num_spikes_wholetrial = length(spike_times_wholetrial);
                    poi_start = start_times(1);
                    poi_stop = stop_times(end);
                    spike_times_poi = spike_times_wholetrial(spike_times_wholetrial >= poi_start & spike_times_wholetrial <= poi_stop);
                    num_spikes_poi = length(spike_times_poi);
                    if num_spikes_poi == 0
                        return
                    else
                        total_poi = poi_stop - poi_start;
                        mean_ISI_poi = total_poi/num_spikes_poi;

                        last_spiketime_before_first_bin = max(spike_times_wholetrial(find(spike_times_wholetrial < poi_start)));
                        if isempty(last_spiketime_before_first_bin),
                            last_spiketime_before_first_bin = min(spike_times_poi(1) - mean_ISI_poi, poi_start);
                        end

```



```
                    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
```

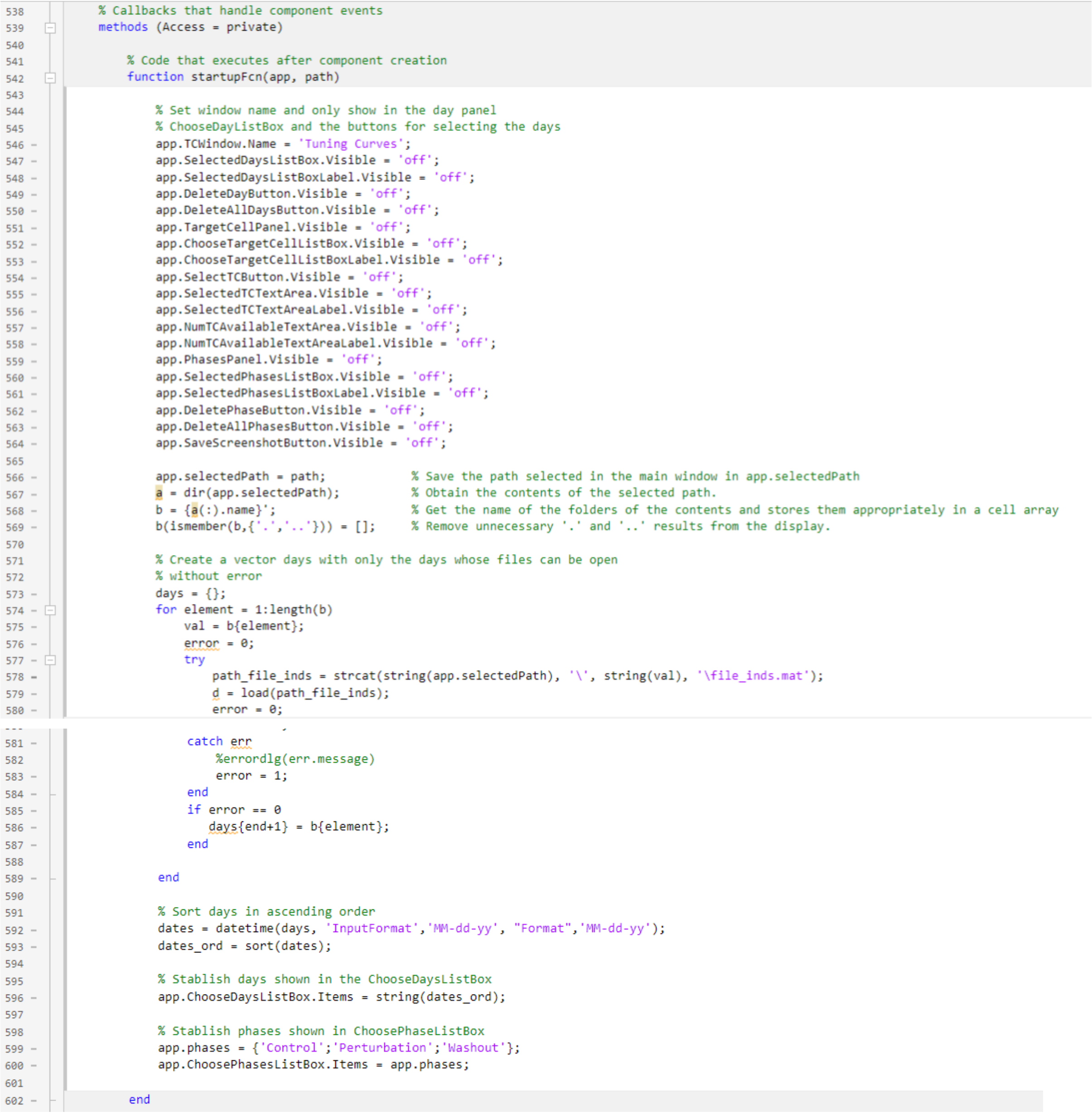


```
        function fit=compute_cosine_fit(alltargs,allrates)

                [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,bint,resid,rint,stats] = regress(allrates(:,i),alltargs,.95);
                        fit.bs(i,:)=b;
                        fit.b0s(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,r]=cart2pol(b(2),b(3));
                            fit.thetas(i,1)=theta*180/pi;
                        else
                            [theta,phi,r]=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.b0s(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;
```



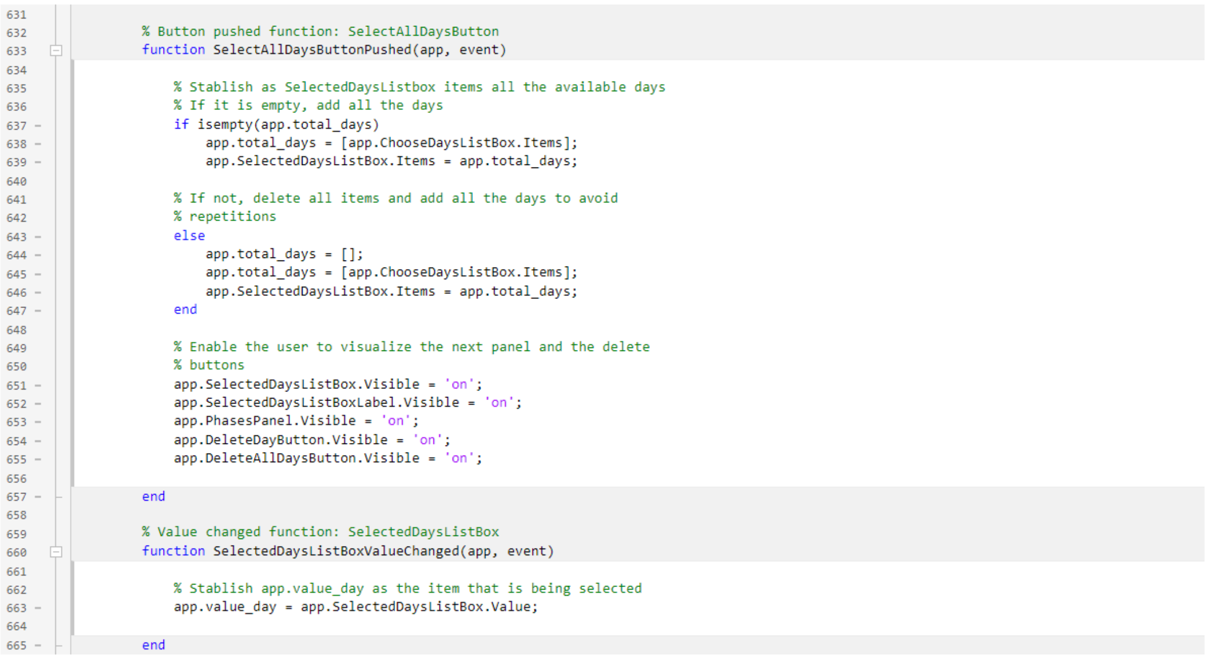
```
                        else
                            fit.thetas(i,1)=NaN; fit.phis(i,1)=NaN;
                        end
                        fit.r2s(i,1)=NaN;
                        fit.pvals(i,1)=NaN;
                        fit.coefferr(:,:,i)=NaN*ones(ndim+1);
                        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

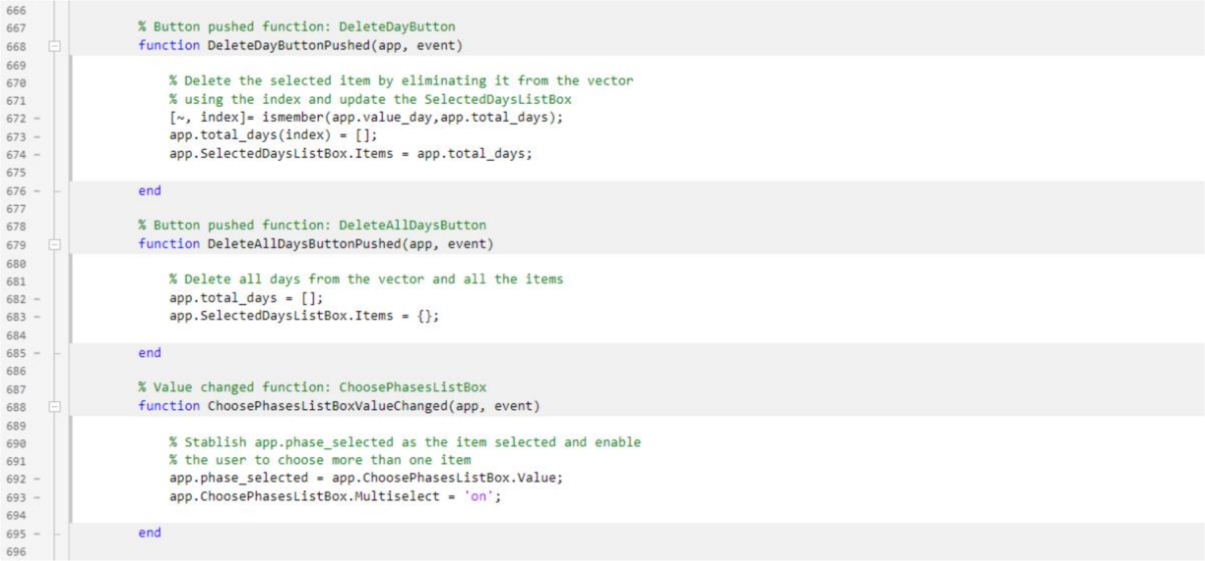


```
    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app, path)

            % Set window name and only show in the day panel
            % ChooseDayListBox and the buttons for selecting the days
            app.TCWindow.Name = 'Tuning Curves';
            app.SelectedDaysListBox.Visible = 'off';
            app.SelectedDaysListBoxLabel.Visible = 'off';
            app.DeleteDayButton.Visible = 'off';
            app.DeleteAllDaysButton.Visible = 'off';
            app.TargetCellPanel.Visible = 'off';
            app.ChooseTargetCellListBox.Visible = 'off';
            app.ChooseTargetCellListBoxLabel.Visible = 'off';
            app.SelectTCButton.Visible = 'off';
            app.SelectedTCTextArea.Visible = 'off';
            app.SelectedTCTextAreaLabel.Visible = 'off';
            app.NumTCAvailableTextArea.Visible = 'off';
            app.NumTCAvailableTextAreaLabel.Visible = 'off';
            app.PhasesPanel.Visible = 'off';
            app.SelectedPhasesListBox.Visible = 'off';
            app.SelectedPhasesListBoxLabel.Visible = 'off';
            app.DeletePhaseButton.Visible = 'off';
            app.DeleteAllPhasesButton.Visible = 'off';
            app.SaveScreenshotButton.Visible = 'off';

            app.selectedPath = path;            % Save the path selected in the main window in app.selectedPath
            a = dir(app.selectedPath);          % Obtain the contents of the selected path.
            b = {a(:).name}';                   % Get the name of the folders of the contents and stores them appropriately in a cell array
            b(ismember(b,{'.','..'})) = [];     % Remove unnecessary '.' and '..' results from the display.

            % Create a vector days with only the days whose files can be open
            % without error
            days = {};
            for element = 1:length(b)
                val = b{element};
                error = 0;
                try
                    path_file_inds = strcat(string(app.selectedPath), '\', string(val), '\file_inds.mat');
                    d = load(path_file_inds);
                    error = 0;
                catch err
                    %errordlg(err.message)
                    error = 1;
                end
                if error == 0
                    days{end+1} = b{element};
                end

            end

            % Sort days in ascending order
            dates = datetime(days, 'InputFormat','MM-dd-yy', "Format",'MM-dd-yy');
            dates_ord = sort(dates);

            % Stablish days shown in the ChooseDaysListBox
            app.ChooseDaysListBox.Items = string(dates_ord);

            % Stablish phases shown in ChoosePhaseListBox
            app.phases = {'Control';'Perturbation';'Washout'};
            app.ChoosePhasesListBox.Items = app.phases;

        end
```

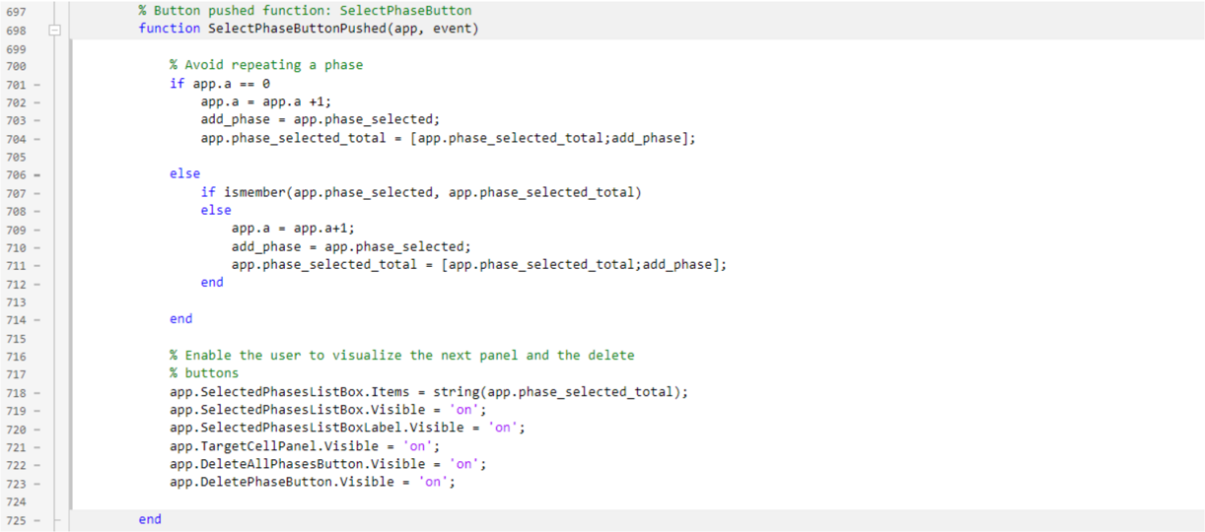
Callback function



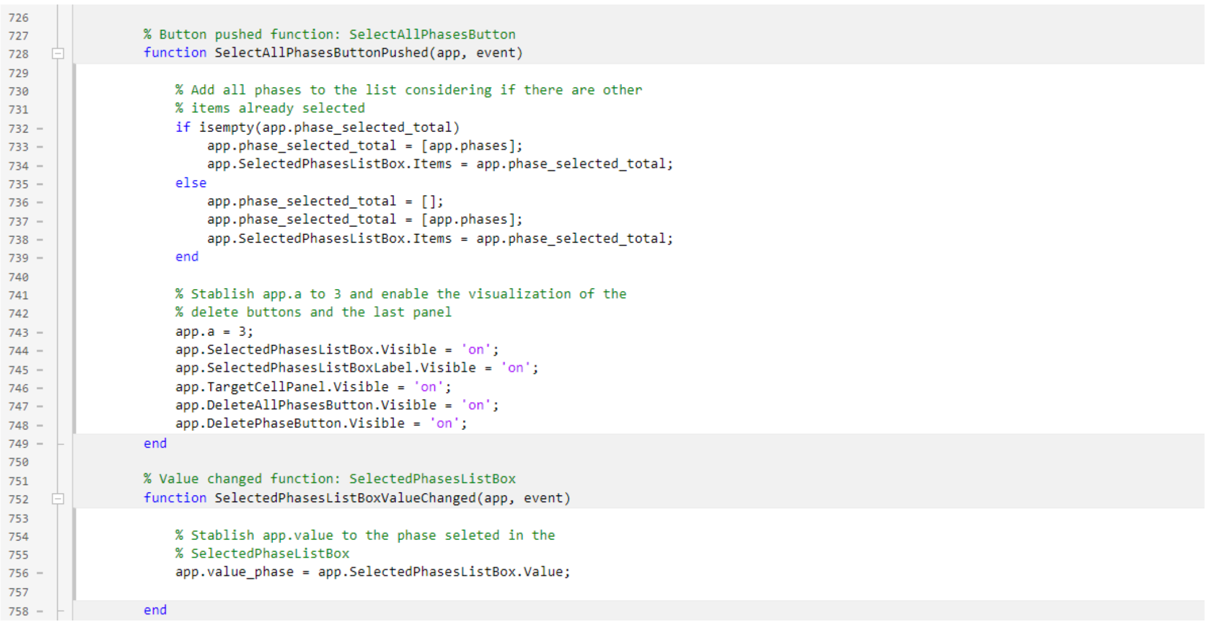
```
        % Button pushed function: SelectAllDaysButton
        function SelectAllDaysButtonPushed(app, event)

            % Stablish as SelectedDaysListbox items all the available days
            % If it is empty, add all the days
            if isempty(app.total_days)
                app.total_days = [app.ChooseDaysListBox.Items];
                app.SelectedDaysListBox.Items = app.total_days;

            % If not, delete all items and add all the days to avoid
            % repetitions
            else
                app.total_days = [];
                app.total_days = [app.ChooseDaysListBox.Items];
                app.SelectedDaysListBox.Items = app.total_days;
            end

            % Enable the user to visualize the next panel and the delete
            % buttons
            app.SelectedDaysListBox.Visible = 'on';
            app.SelectedDaysListBoxLabel.Visible = 'on';
            app.PhasesPanel.Visible = 'on';
            app.DeleteDayButton.Visible = 'on';
            app.DeleteAllDaysButton.Visible = 'on';

        end

        % Value changed function: SelectedDaysListBox
        function SelectedDaysListBoxValueChanged(app, event)

            % Stablish app.value_day as the item that is being selected
            app.value_day = app.SelectedDaysListBox.Value;

        end
```



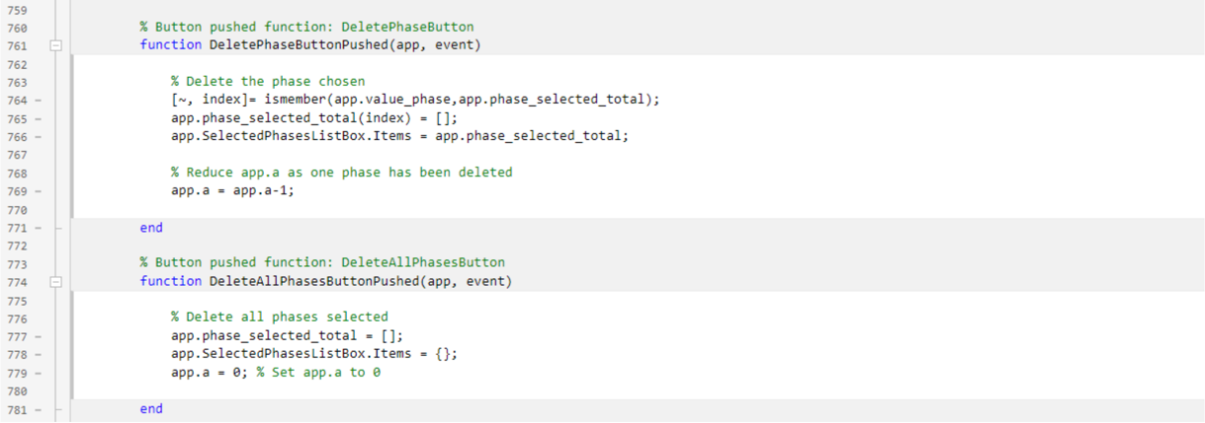
```
        % Button pushed function: DeleteDayButton
        function DeleteDayButtonPushed(app, event)

            % Delete the selected item by eliminating it from the vector
            % using the index and update the SelectedDaysListBox
            [~, index]= ismember(app.value_day,app.total_days);
            app.total_days(index) = [];
            app.SelectedDaysListBox.Items = app.total_days;

        end

        % Button pushed function: DeleteAllDaysButton
        function DeleteAllDaysButtonPushed(app, event)

            % Delete all days from the vector and all the items
            app.total_days = [];
            app.SelectedDaysListBox.Items = {};

        end

        % Value changed function: ChoosePhasesListBox
        function ChoosePhasesListBoxValueChanged(app, event)

            % Stablish app.phase_selected as the item selected and enable
            % the user to choose more than one item
            app.phase_selected = app.ChoosePhasesListBox.Value;
            app.ChoosePhasesListBox.Multiselect = 'on';

        end

```



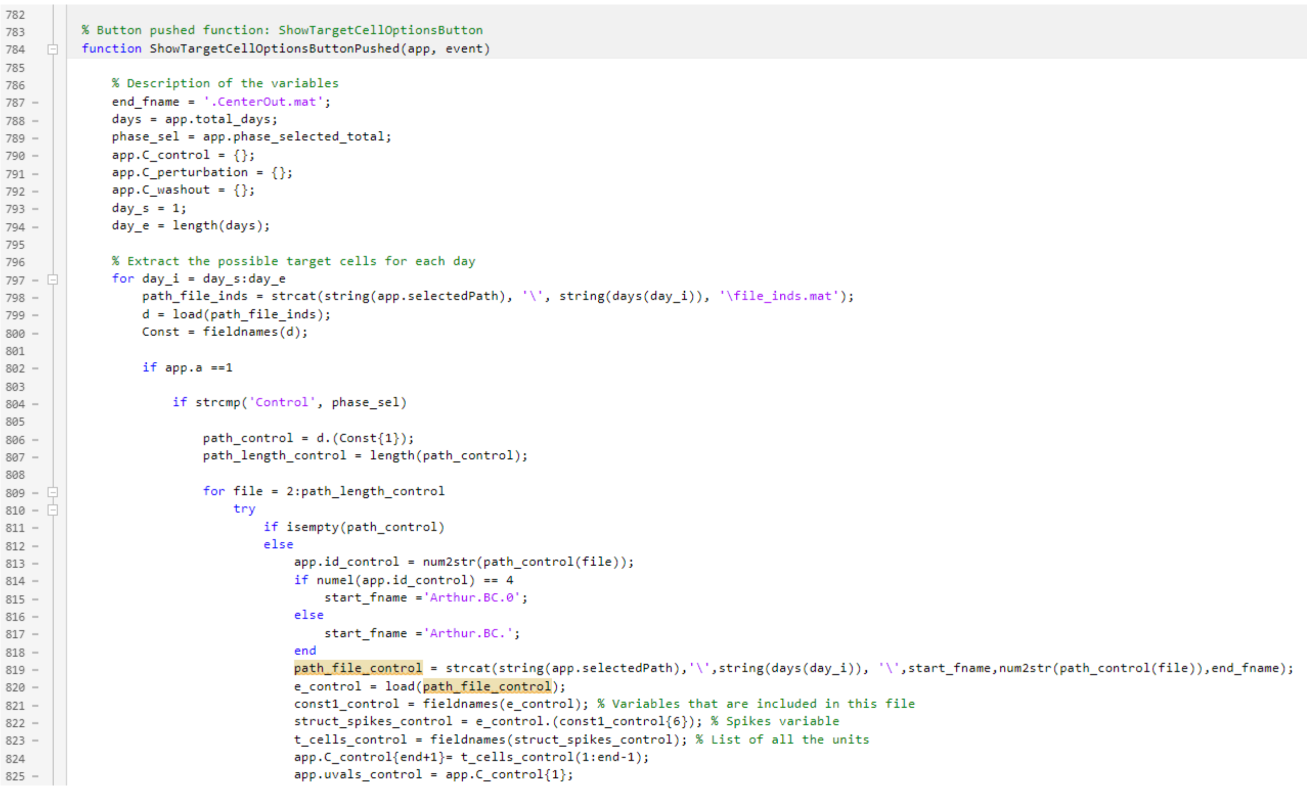
```
        % Button pushed function: SelectPhaseButton
        function SelectPhaseButtonPushed(app, event)

            % Avoid repeating a phase
            if app.a == 0
                app.a = app.a +1;
                add_phase = app.phase_selected;
                app.phase_selected_total = [app.phase_selected_total;add_phase];

            else
                if ismember(app.phase_selected, app.phase_selected_total)
                else
                    app.a = app.a+1;
                    add_phase = app.phase_selected;
                    app.phase_selected_total = [app.phase_selected_total;add_phase];
                end

            end

            % Enable the user to visualize the next panel and the delete
            % buttons
            app.SelectedPhasesListBox.Items = string(app.phase_selected_total);
            app.SelectedPhasesListBox.Visible = 'on';
            app.SelectedPhasesListBoxLabel.Visible = 'on';
            app.TargetCellPanel.Visible = 'on';
            app.DeleteAllPhasesButton.Visible = 'on';
            app.DeletePhaseButton.Visible = 'on';

        end
```



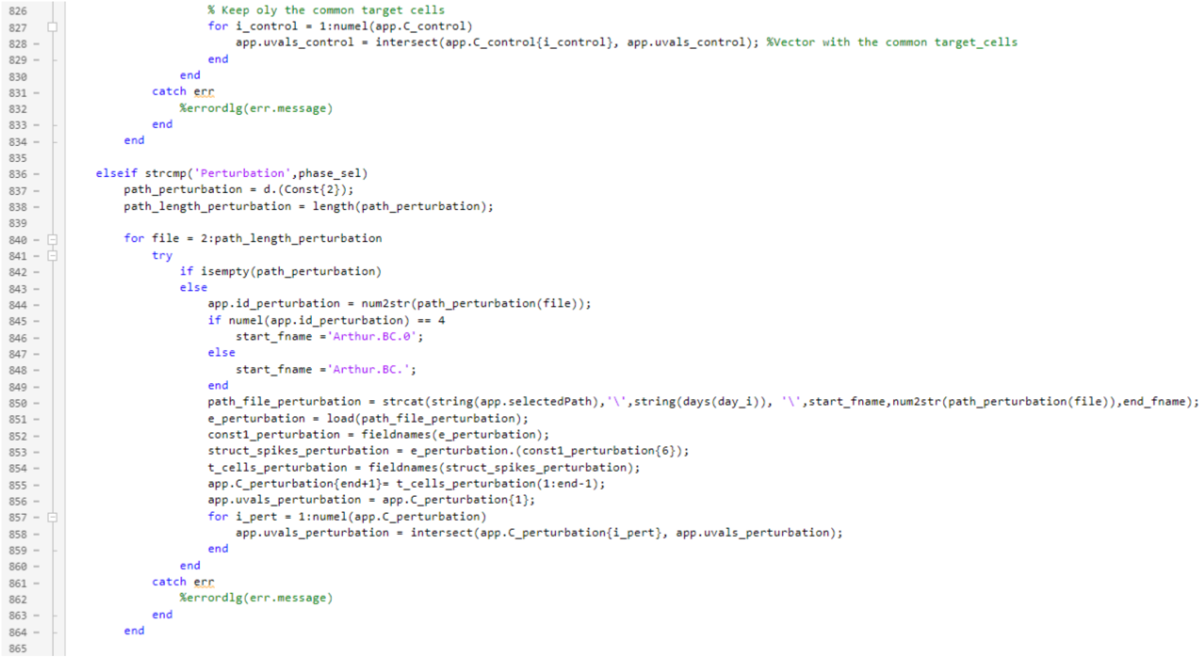
```

        % Button pushed function: SelectAllPhasesButton
        function SelectAllPhasesButtonPushed(app, event)

            % Add all phases to the list considering if there are other
            % items already selected
            if isempty(app.phase_selected_total)
                app.phase_selected_total = [app.phases];
                app.SelectedPhasesListBox.Items = app.phase_selected_total;
            else
                app.phase_selected_total = [];
                app.phase_selected_total = [app.phases];
                app.SelectedPhasesListBox.Items = app.phase_selected_total;
            end

            % Stablish app.a to 3 and enable the visualization of the
            % delete buttons and the last panel
            app.a = 3;
            app.SelectedPhasesListBox.Visible = 'on';
            app.SelectedPhasesListBoxLabel.Visible = 'on';
            app.TargetCellPanel.Visible = 'on';
            app.DeleteAllPhasesButton.Visible = 'on';
            app.DeletePhaseButton.Visible = 'on';
        end

        % Value changed function: SelectedPhasesListBox
        function SelectedPhasesListBoxValueChanged(app, event)

            % Stablish app.value to the phase seleted in the
            % SelectedPhaseListBox
            app.value_phase = app.SelectedPhasesListBox.Value;

        end
```



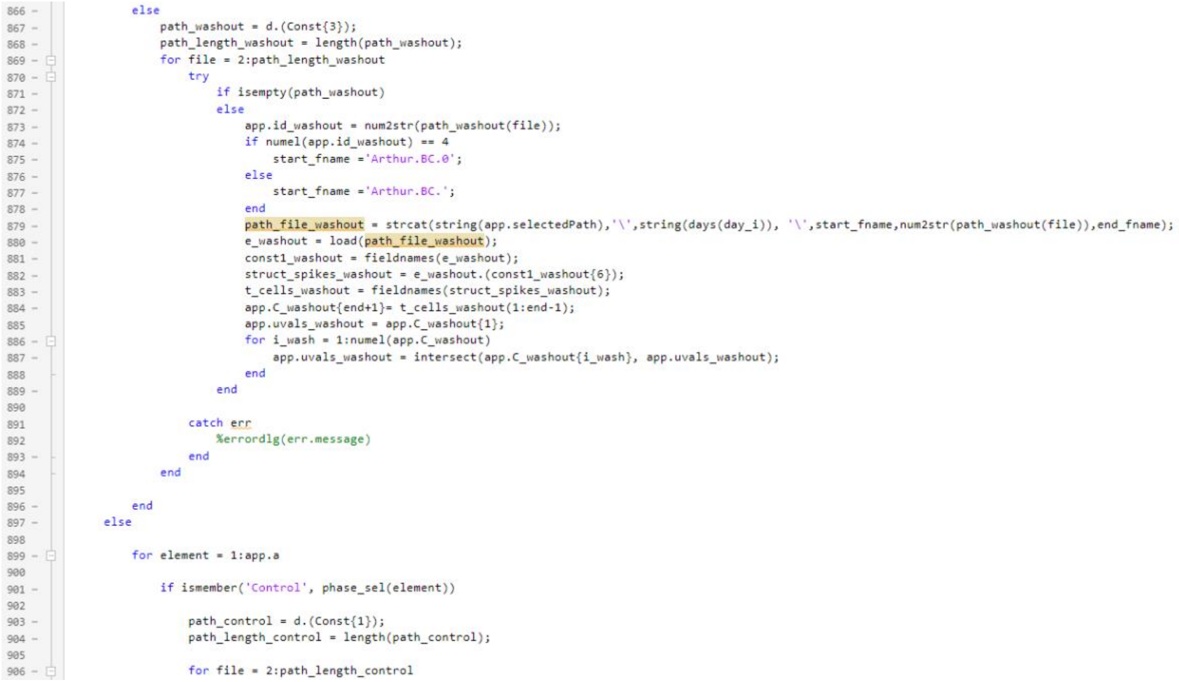
```

        % Button pushed function: DeletePhaseButton
        function DeletePhaseButtonPushed(app, event)

            % Delete the phase chosen
            [~, index]= ismember(app.value_phase,app.phase_selected_total);
            app.phase_selected_total(index) = [];
            app.SelectedPhasesListBox.Items = app.phase_selected_total;

            % Reduce app.a as one phase has been deleted
            app.a = app.a-1;

        end

        % Button pushed function: DeleteAllPhasesButton
        function DeleteAllPhasesButtonPushed(app, event)

            % Delete all phases selected
            app.phase_selected_total = [];
            app.SelectedPhasesListBox.Items = {};
            app.a = 0; % Set app.a to 0

        end
```



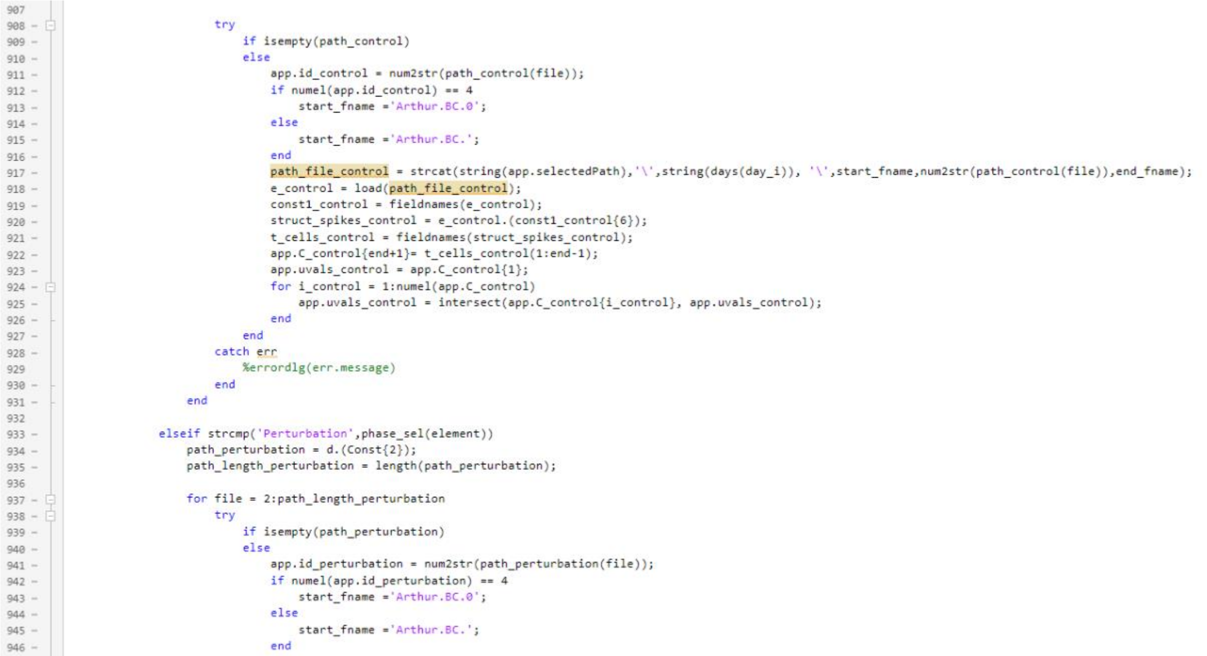
```

        % Button pushed function: ShowTargetCellOptionsButton
        function ShowTargetCellOptionsButtonPushed(app, event)

            % Description of the variables
            end_fname = '.CenterOut.mat';
            days = app.total_days;
            phase_sel = app.phase_selected_total;
            app.C_control = {};
            app.C_perturbation = {};
            app.C_washout = {};
            day_s = 1;
            day_e = length(days);

            % Extract the possible target cells for each day
            for day_i = day_s:day_e
                path_file_inds = strcat(string(app.selectedPath), '\', string(days(day_i)), '\file_inds.mat');
                d = load(path_file_inds);
                Const = fieldnames(d);

                if app.a ==1

                    if strcmp('Control', phase_sel)

                        path_control = d.(Const{1});
                        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_i)), '\',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{1};
```



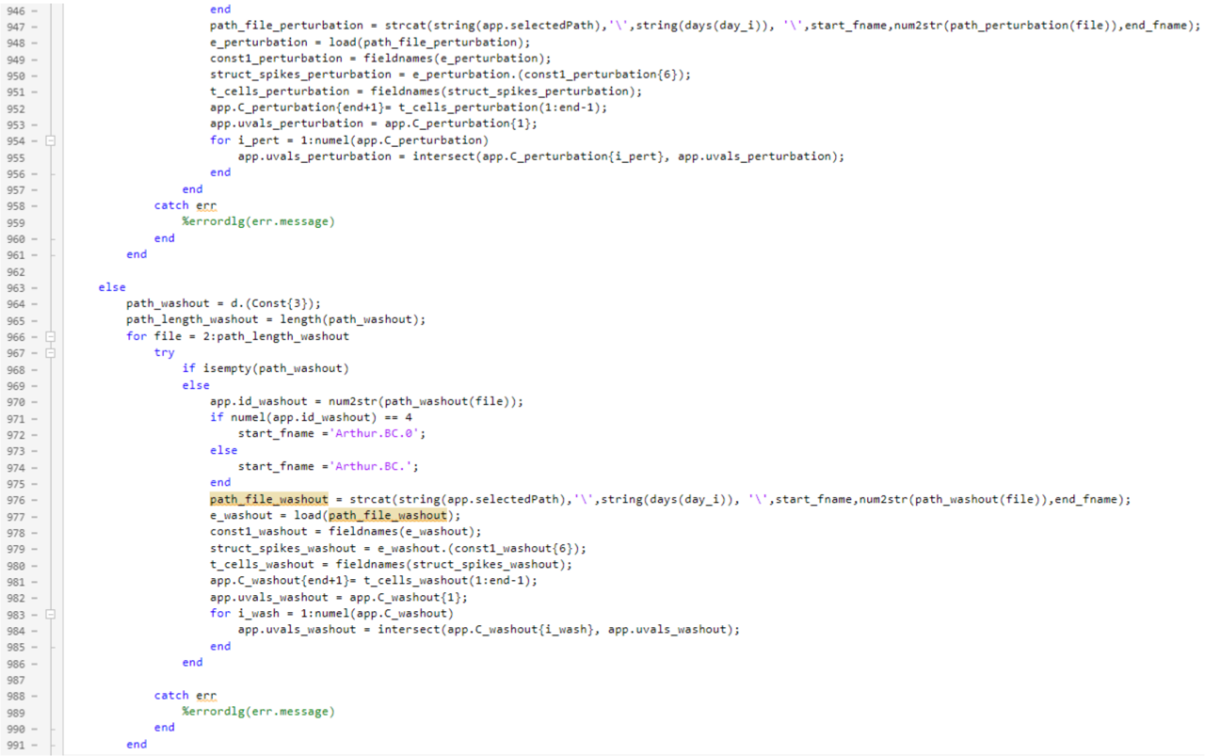
```
                % Keep oly the common target cells
                for i_control = 1:numel(app.C_control)
                    app.uvals_control = intersect(app.C_control{i_control}, app.uvals_control); %Vector with the common target_cells
                end
            end
        catch err
            %errordlg(err.message)
        end
    end

elseif strcmp('Perturbation',phase_sel)
    path_perturbation = d.(Const{2});
    path_length_perturbation = length(path_perturbation);

    for file = 2:path_length_perturbation
        try
            if isempty(path_perturbation)
            else
                app.id_perturbation = num2str(path_perturbation(file));
                if numel(app.id_perturbation) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
                path_file_perturbation = strcat(string(app.selectedPath),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
                e_perturbation = load(path_file_perturbation);
                const1_perturbation = fieldnames(e_perturbation);
                struct_spikes_perturbation = e_perturbation.(const1_perturbation{6});
                t_cells_perturbation = fieldnames(struct_spikes_perturbation);
                app.C_perturbation{end+1}= t_cells_perturbation(1:end-1);
                app.uvals_perturbation = app.C_perturbation{1};
                for i_pert = 1:numel(app.C_perturbation)
                    app.uvals_perturbation = intersect(app.C_perturbation{i_pert}, app.uvals_perturbation);
                end
            end
        catch err
            %errordlg(err.message)
        end
    end

```



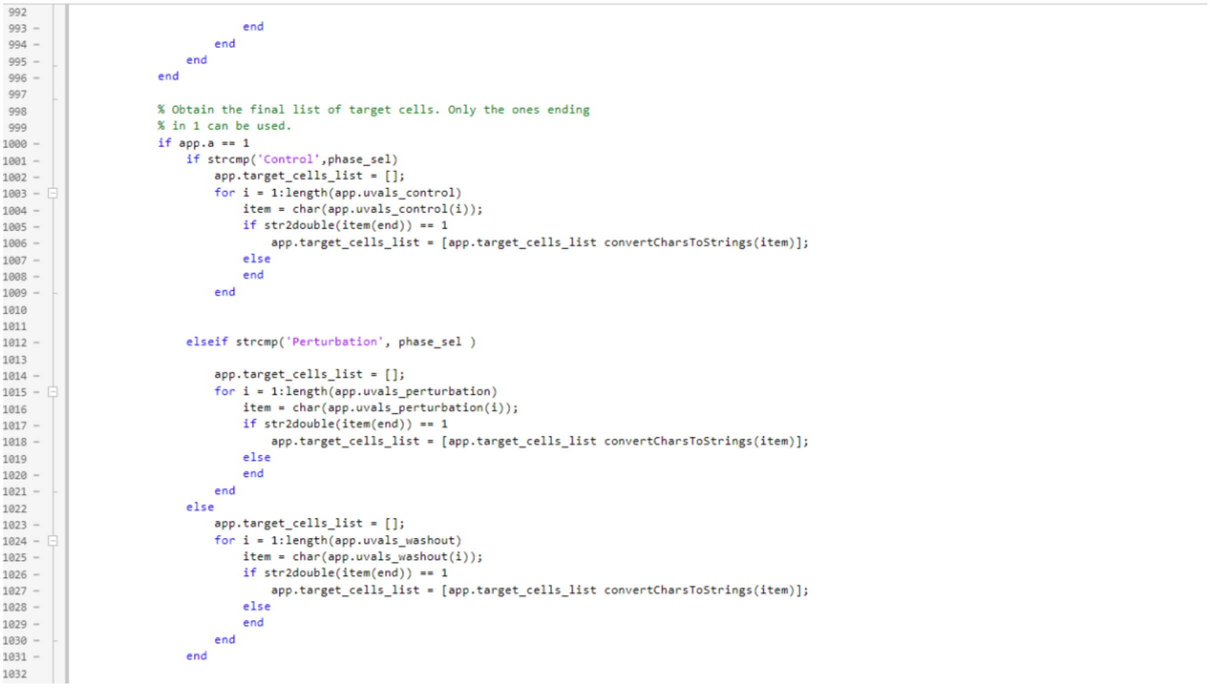
```
    else
        path_washout = d.(Const{3});
        path_length_washout = length(path_washout);
        for file = 2:path_length_washout
            try
                if isempty(path_washout)
                else
                    app.id_washout = num2str(path_washout(file));
                    if numel(app.id_washout) == 4
                        start_fname ='Arthur.BC.0';
                    else
                        start_fname ='Arthur.BC.';
                    end
                    path_file_washout = strcat(string(app.selectedPath),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
                    e_washout = load(path_file_washout);
                    const1_washout = fieldnames(e_washout);
                    struct_spikes_washout = e_washout.(const1_washout{6});
                    t_cells_washout = fieldnames(struct_spikes_washout);
                    app.C_washout{end+1}= t_cells_washout(1:end-1);
                    app.uvals_washout = app.C_washout{1};
                    for i_wash = 1:numel(app.C_washout)
                        app.uvals_washout = intersect(app.C_washout{i_wash}, app.uvals_washout);
                    end
                end

            catch err
                %errordlg(err.message)
            end
        end

    end
else

    for element = 1:app.a

        if ismember('Control', phase_sel(element))

            path_control = d.(Const{1});
            path_length_control = length(path_control);

            for file = 2:path_length_control
```



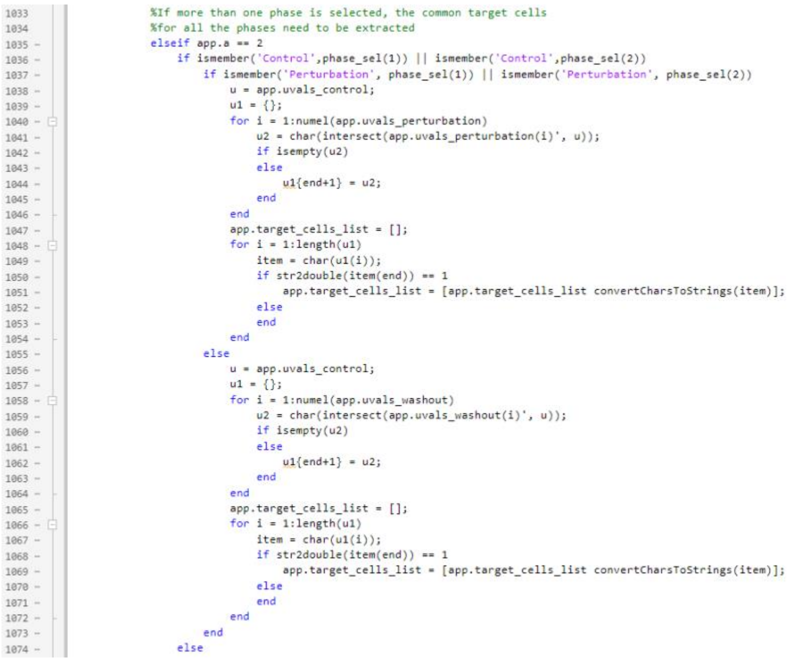
```

                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_i)), '\',start_fname,num2str(path_control(file)),end_fname);
                        e_control = load(path_file_control);
                        const1_control = fieldnames(e_control);
                        struct_spikes_control = e_control.(const1_control{6});
                        t_cells_control = fieldnames(struct_spikes_control);
                        app.C_control{end+1}= t_cells_control(1:end-1);
                        app.uvals_control = app.C_control{1};
                        for i_control = 1:numel(app.C_control)
                            app.uvals_control = intersect(app.C_control{i_control}, app.uvals_control);
                        end
                    end
                catch err
                    %errordlg(err.message)
                end
            end

        elseif strcmp('Perturbation',phase_sel(element))
            path_perturbation = d.(Const{2});
            path_length_perturbation = length(path_perturbation);

            for file = 2:path_length_perturbation
                try
                    if isempty(path_perturbation)
                    else
                        app.id_perturbation = num2str(path_perturbation(file));
                        if numel(app.id_perturbation) == 4
                            start_fname ='Arthur.BC.0';
                        else
                            start_fname ='Arthur.BC.';
                        end
```



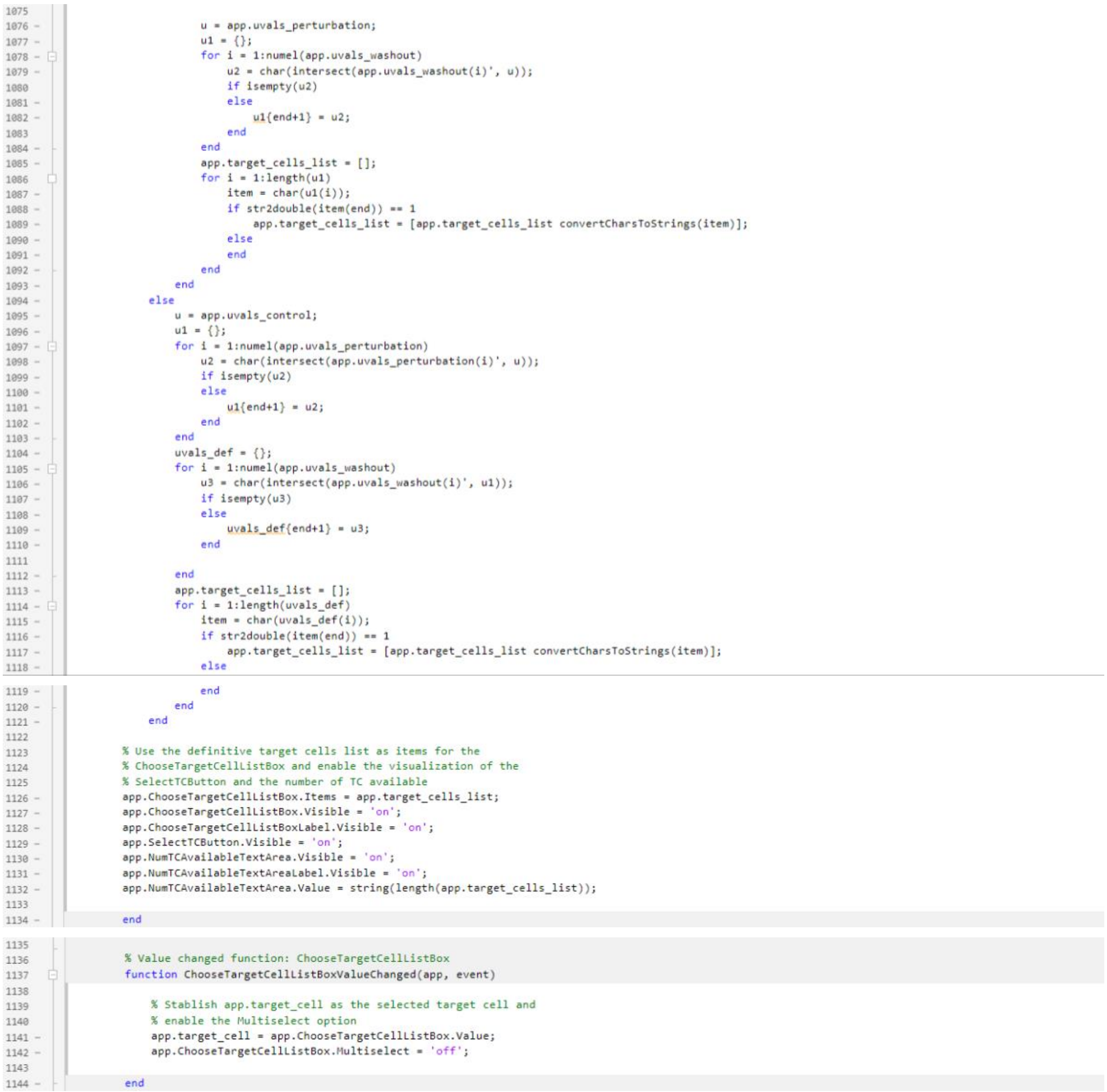
```
                        end
                        path_file_perturbation = strcat(string(app.selectedPath),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
                        e_perturbation = load(path_file_perturbation);
                        const1_perturbation = fieldnames(e_perturbation);
                        struct_spikes_perturbation = e_perturbation.(const1_perturbation{6});
                        t_cells_perturbation = fieldnames(struct_spikes_perturbation);
                        app.C_perturbation{end+1}= t_cells_perturbation(1:end-1);
                        app.uvals_perturbation = app.C_perturbation{1};
                        for i_pert = 1:numel(app.C_perturbation)
                            app.uvals_perturbation = intersect(app.C_perturbation{i_pert}, app.uvals_perturbation);
                        end
                    end
                catch err
                    %errordlg(err.message)
                end
            end

        else
            path_washout = d.(Const{3});
            path_length_washout = length(path_washout);
            for file = 2:path_length_washout
                try
                    if isempty(path_washout)
                    else
                        app.id_washout = num2str(path_washout(file));
                        if numel(app.id_washout) == 4
                            start_fname ='Arthur.BC.0';
                        else
                            start_fname ='Arthur.BC.';
                        end
                        path_file_washout = strcat(string(app.selectedPath),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
                        e_washout = load(path_file_washout);
                        const1_washout = fieldnames(e_washout);
                        struct_spikes_washout = e_washout.(const1_washout{6});
                        t_cells_washout = fieldnames(struct_spikes_washout);
                        app.C_washout{end+1}= t_cells_washout(1:end-1);
                        app.uvals_washout = app.C_washout{1};
                        for i_wash = 1:numel(app.C_washout)
                            app.uvals_washout = intersect(app.C_washout{i_wash}, app.uvals_washout);
                        end
                    end

                catch err
                    %errordlg(err.message)
                end
            end
```



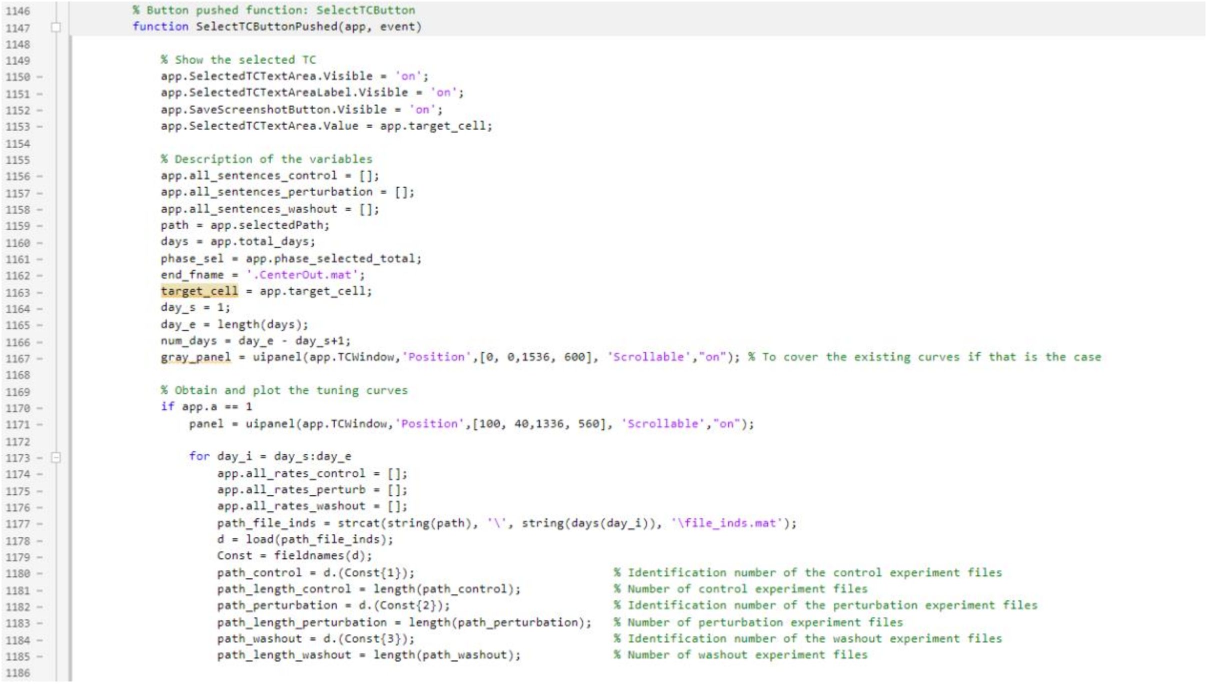
```
                end
            end
        end
    end

    % Obtain the final list of target cells. Only the ones ending
    % in 1 can be used.
    if app.a == 1
        if strcmp('Control',phase_sel)
            app.target_cells_list = [];
            for i = 1:length(app.uvals_control)
                item = char(app.uvals_control(i));
                if str2double(item(end)) == 1
                    app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
                else
                end
            end


        elseif strcmp('Perturbation', phase_sel )

            app.target_cells_list = [];
            for i = 1:length(app.uvals_perturbation)
                item = char(app.uvals_perturbation(i));
                if str2double(item(end)) == 1
                    app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
                else
                end
            end
        else
            app.target_cells_list = [];
            for i = 1:length(app.uvals_washout)
                item = char(app.uvals_washout(i));
                if str2double(item(end)) == 1
                    app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
                else
                end
            end
        end

```



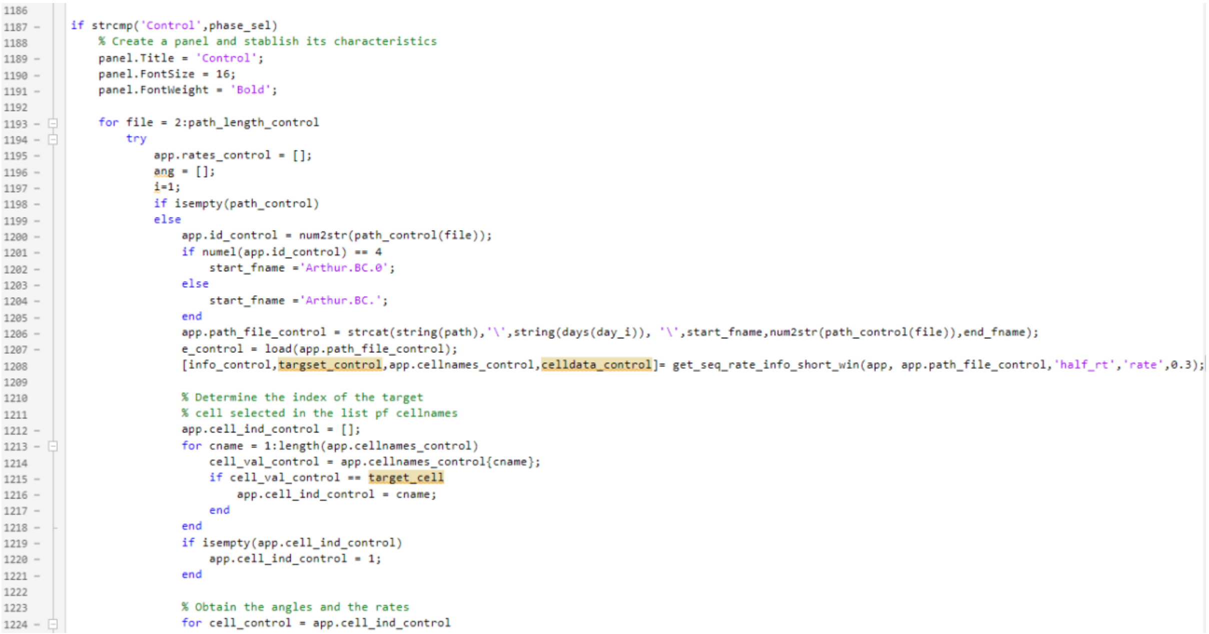
```
    %If more than one phase is selected, the common target cells
    %for all the phases need to be extracted
    elseif app.a == 2
        if ismember('Control',phase_sel(1)) || ismember('Control',phase_sel(2))
            if ismember('Perturbation', phase_sel(1)) || ismember('Perturbation', phase_sel(2))
                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 convertCharsToStrings(item)];
                    else
                    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 convertCharsToStrings(item)];
                    else
                    end
                end
            end
        else
```



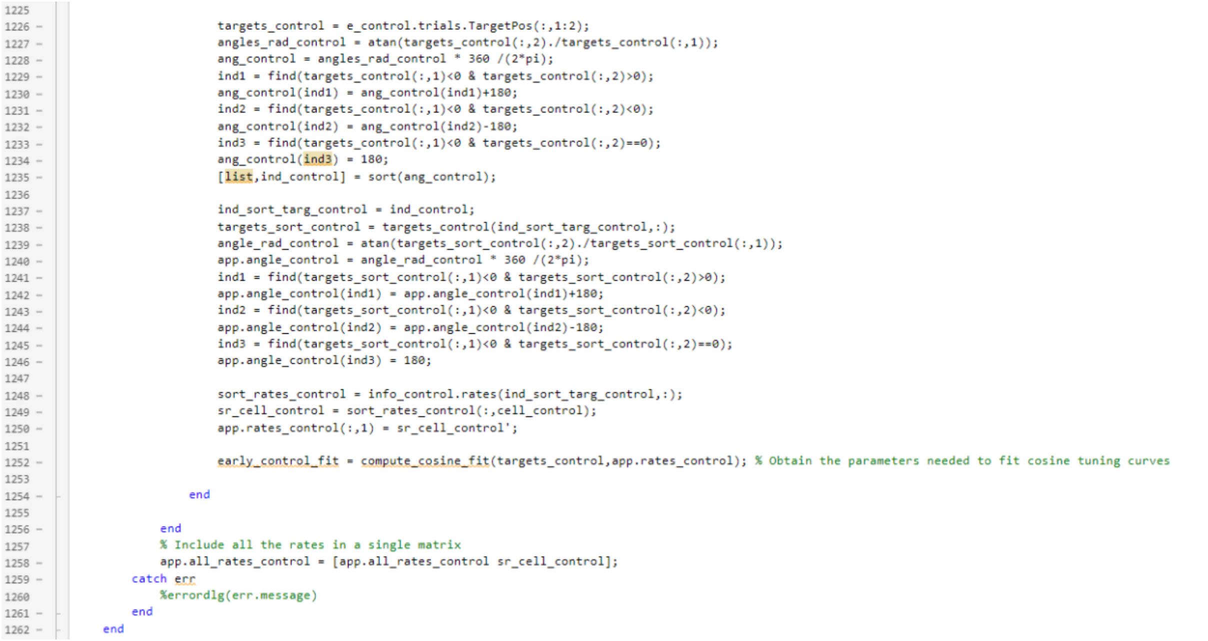
```

                u = app.uvals_perturbation;
                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 convertCharsToStrings(item)];
                    else
                    end
                end
            end
        else
            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
            uvals_def = {};
            for i = 1:numel(app.uvals_washout)
                u3 = char(intersect(app.uvals_washout(i)', u1));
                if isempty(u3)
                else
                    uvals_def{end+1} = u3;
                end

            end
            app.target_cells_list = [];
            for i = 1:length(uvals_def)
                item = char(uvals_def(i));
                if str2double(item(end)) == 1
                    app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
                else
                end
            end
        end

    % Use the definitive target cells list as items for the
    % ChooseTargetCellListBox and enable the visualization of the
    % SelectTCButton and the number of TC available
    app.ChooseTargetCellListBox.Items = app.target_cells_list;
    app.ChooseTargetCellListBox.Visible = 'on';
    app.ChooseTargetCellListBoxLabel.Visible = 'on';
    app.SelectTCButton.Visible = 'on';
    app.NumTCAvailableTextArea.Visible = 'on';
    app.NumTCAvailableTextAreaLabel.Visible = 'on';
    app.NumTCAvailableTextArea.Value = string(length(app.target_cells_list));

    end

    % Value changed function: ChooseTargetCellListBox
    function ChooseTargetCellListBoxValueChanged(app, event)

        % Stablish app.target_cell as the selected target cell and
        % enable the Multiselect option
        app.target_cell = app.ChooseTargetCellListBox.Value;
        app.ChooseTargetCellListBox.Multiselect = 'off';

    end
```



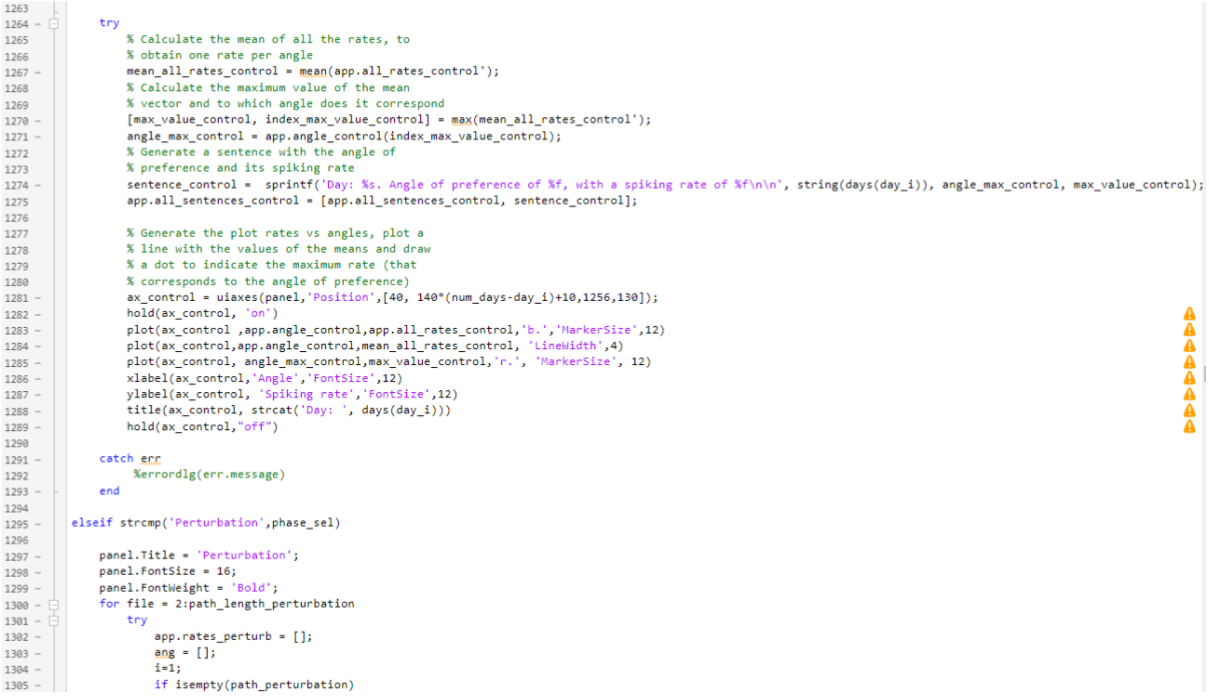
```
        % Button pushed function: SelectTCButton
        function SelectTCButtonPushed(app, event)

            % Show the selected TC
            app.SelectedTCTextArea.Visible = 'on';
            app.SelectedTCTextAreaLabel.Visible = 'on';
            app.SaveScreenshotButton.Visible = 'on';
            app.SelectedTCTextArea.Value = app.target_cell;

            % Description of the variables
            app.all_sentences_control = [];
            app.all_sentences_perturbation = [];
            app.all_sentences_washout = [];
            path = app.selectedPath;
            days = app.total_days;
            phase_sel = app.phase_selected_total;
            end_fname = '.CenterOut.mat';
            target_cell = app.target_cell;
            day_s = 1;
            day_e = length(days);
            num_days = day_e - day_s+1;
            gray_panel = uipanel(app.TCWindow,'Position',[0, 0,1536, 600], 'Scrollable',"on"); % To cover the existing curves if that is the case

            % Obtain and plot the tuning curves
            if app.a == 1
                panel = uipanel(app.TCWindow,'Position',[100, 40,1336, 560], 'Scrollable',"on");

                for day_i = day_s:day_e
                    app.all_rates_control = [];
                    app.all_rates_perturb = [];
                    app.all_rates_washout = [];
                    path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
                    d = load(path_file_inds);
                    Const = fieldnames(d);
                    path_control = d.(Const{1});                            % Identification number of the control experiment files
                    path_length_control = length(path_control);             % Number of control experiment files
                    path_perturbation = d.(Const{2});                       % Identification number of the perturbation experiment files
                    path_length_perturbation = length(path_perturbation);   % Number of perturbation experiment files
                    path_washout = d.(Const{3});                            % Identification number of the washout experiment files
                    path_length_washout = length(path_washout);             % Number of washout experiment files

```



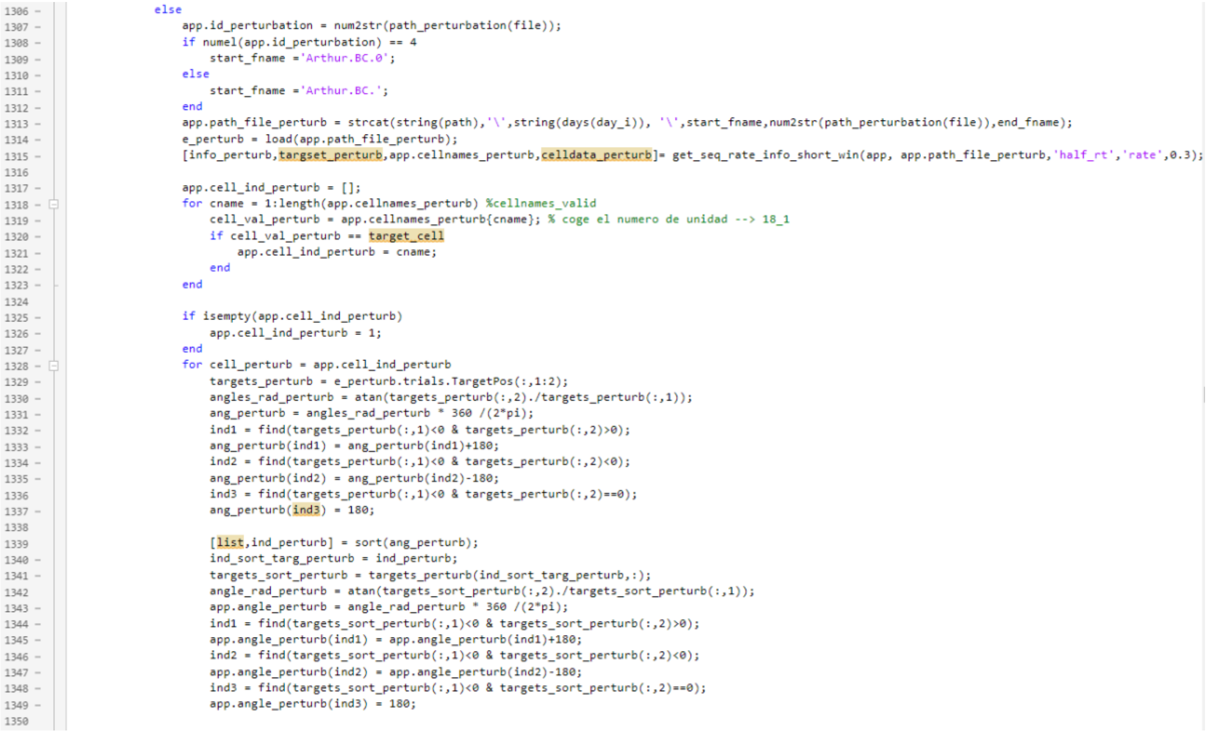
```

if strcmp('Control',phase_sel)
    % Create a panel and stablish its characteristics
    panel.Title = 'Control';
    panel.FontSize = 16;
    panel.FontWeight = 'Bold';

    for file = 2:path_length_control
        try
            app.rates_control = [];
            ang = [];
            i=1;
            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
                app.path_file_control = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_control(file)),end_fname);
                e_control = load(app.path_file_control);
                [info_control,targset_control,app.cellnames_control,celldata_control]= get_seq_rate_info_short_win(app, app.path_file_control,'half_rt','rate',0.3);

                % Determine the index of the target
                % cell selected in the list pf cellnames
                app.cell_ind_control = [];
                for cname = 1:length(app.cellnames_control)
                    cell_val_control = app.cellnames_control{cname};
                    if cell_val_control == target_cell
                        app.cell_ind_control = cname;
                    end
                end
                if isempty(app.cell_ind_control)
                    app.cell_ind_control = 1;
                end

                % Obtain the angles and the rates
                for cell_control = app.cell_ind_control
```



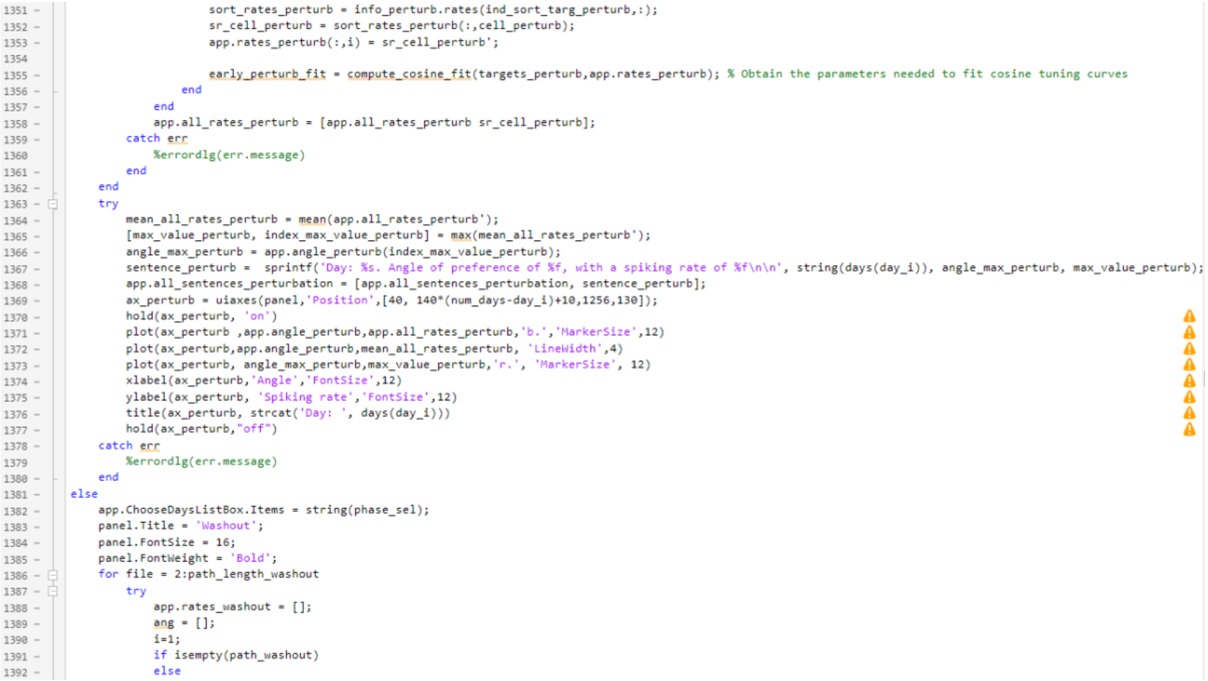
```
                    targets_control = e_control.trials.TargetPos(:,1:2);
                    angles_rad_control = atan(targets_control(:,2)./targets_control(:,1));
                    ang_control = angles_rad_control * 360 /(2*pi);
                    ind1 = find(targets_control(:,1)<0 & targets_control(:,2)>0);
                    ang_control(ind1) = ang_control(ind1)+180;
                    ind2 = find(targets_control(:,1)<0 & targets_control(:,2)<0);
                    ang_control(ind2) = ang_control(ind2)-180;
                    ind3 = find(targets_control(:,1)<0 & targets_control(:,2)==0);
                    ang_control(ind3) = 180;
                    [list,ind_control] = sort(ang_control);

                    ind_sort_targ_control = ind_control;
                    targets_sort_control = targets_control(ind_sort_targ_control,:);
                    angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
                    app.angle_control = angle_rad_control * 360 /(2*pi);
                    ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
                    app.angle_control(ind1) = app.angle_control(ind1)+180;
                    ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
                    app.angle_control(ind2) = app.angle_control(ind2)-180;
                    ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
                    app.angle_control(ind3) = 180;

                    sort_rates_control = info_control.rates(ind_sort_targ_control,:);
                    sr_cell_control = sort_rates_control(:,cell_control);
                    app.rates_control(:,1) = sr_cell_control';

                    early_control_fit = compute_cosine_fit(targets_control,app.rates_control); % Obtain the parameters needed to fit cosine tuning curves

                end

            end
            % Include all the rates in a single matrix
            app.all_rates_control = [app.all_rates_control sr_cell_control];
        catch err
            %errordlg(err.message)
        end
    end
```



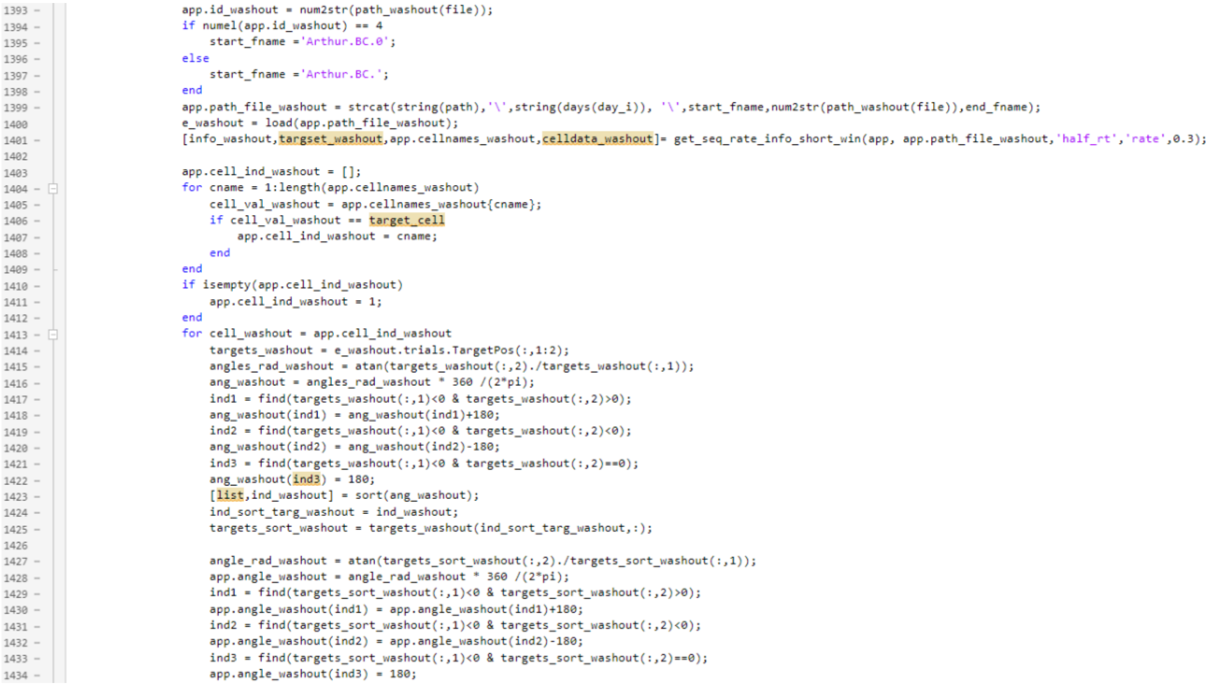
```
    try
        % Calculate the mean of all the rates, to
        % obtain one rate per angle
        mean_all_rates_control = mean(app.all_rates_control');
        % Calculate the maximum value of the mean
        % vector and to which angle does it correspond
        [max_value_control, index_max_value_control] = max(mean_all_rates_control');
        angle_max_control = app.angle_control(index_max_value_control);
        % Generate a sentence with the angle of
        % preference and its spiking rate
        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);
        app.all_sentences_control = [app.all_sentences_control, sentence_control];

        % Generate the plot rates vs angles, plot a
        % line with the values of the means and draw
        % a dot to indicate the maximum rate (that
        % corresponds to the angle of preference)
        ax_control = uiaxes(panel,'Position',[40, 140*(num_days-day_i)+10,1256,130]);
        hold(ax_control, 'on')
        plot(ax_control ,app.angle_control,app.all_rates_control,'b.','MarkerSize',12)
        plot(ax_control,app.angle_control,mean_all_rates_control, 'LineWidth',4)
        plot(ax_control, angle_max_control,max_value_control,'r.', 'MarkerSize', 12)
        xlabel(ax_control,'Angle','FontSize',12)
        ylabel(ax_control, 'Spiking rate','FontSize',12)
        title(ax_control, strcat('Day: ', days(day_i)))
        hold(ax_control,"off")

    catch err
         %errordlg(err.message)
    end

elseif strcmp('Perturbation',phase_sel)

    panel.Title = 'Perturbation';
    panel.FontSize = 16;
    panel.FontWeight = 'Bold';
    for file = 2:path_length_perturbation
        try
            app.rates_perturb = [];
            ang = [];
            i=1;
            if isempty(path_perturbation)
```



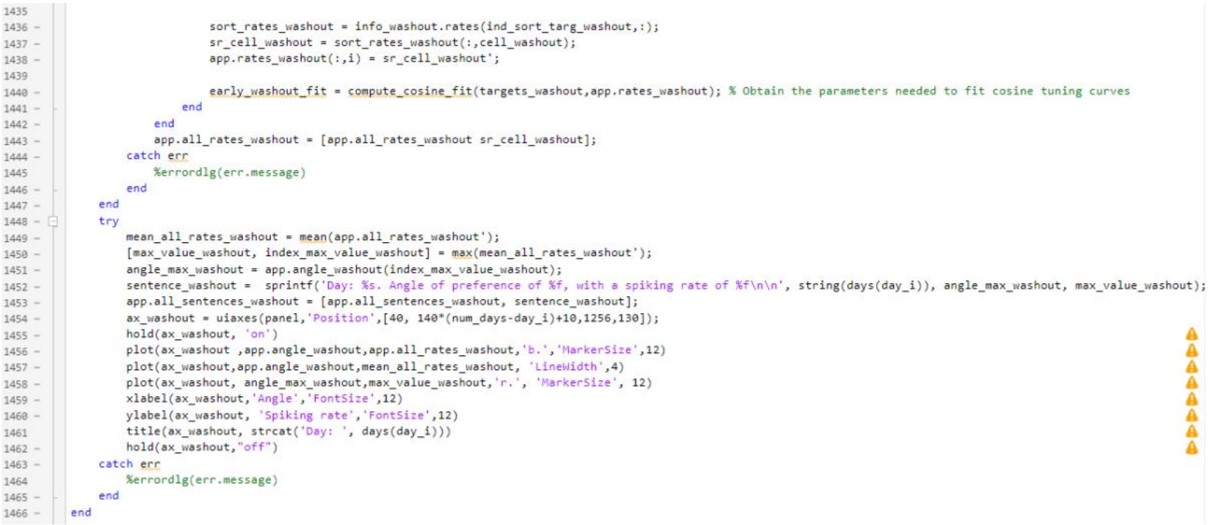
```
            else
                app.id_perturbation = num2str(path_perturbation(file));
                if numel(app.id_perturbation) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
                app.path_file_perturb = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
                e_perturb = load(app.path_file_perturb);
                [info_perturb,targset_perturb,app.cellnames_perturb,celldata_perturb]= get_seq_rate_info_short_win(app, app.path_file_perturb,'half_rt','rate',0.3);

                app.cell_ind_perturb = [];
                for cname = 1:length(app.cellnames_perturb) %cellnames_valid
                    cell_val_perturb = app.cellnames_perturb{cname}; % coge el numero de unidad --> 18_1
                    if cell_val_perturb == target_cell
                        app.cell_ind_perturb = cname;
                    end
                end

                if isempty(app.cell_ind_perturb)
                    app.cell_ind_perturb = 1;
                end
                for cell_perturb = app.cell_ind_perturb
                    targets_perturb = e_perturb.trials.TargetPos(:,1:2);
                    angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
                    ang_perturb = angles_rad_perturb * 360 /(2*pi);
                    ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
                    ang_perturb(ind1) = ang_perturb(ind1)+180;
                    ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
                    ang_perturb(ind2) = ang_perturb(ind2)-180;
                    ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
                    ang_perturb(ind3) = 180;

                    [list,ind_perturb] = sort(ang_perturb);
                    ind_sort_targ_perturb = ind_perturb;
                    targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);
                    angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
                    app.angle_perturb = angle_rad_perturb * 360 /(2*pi);
                    ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
                    app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
                    ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
                    app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
                    ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
                    app.angle_perturb(ind3) = 180;

```



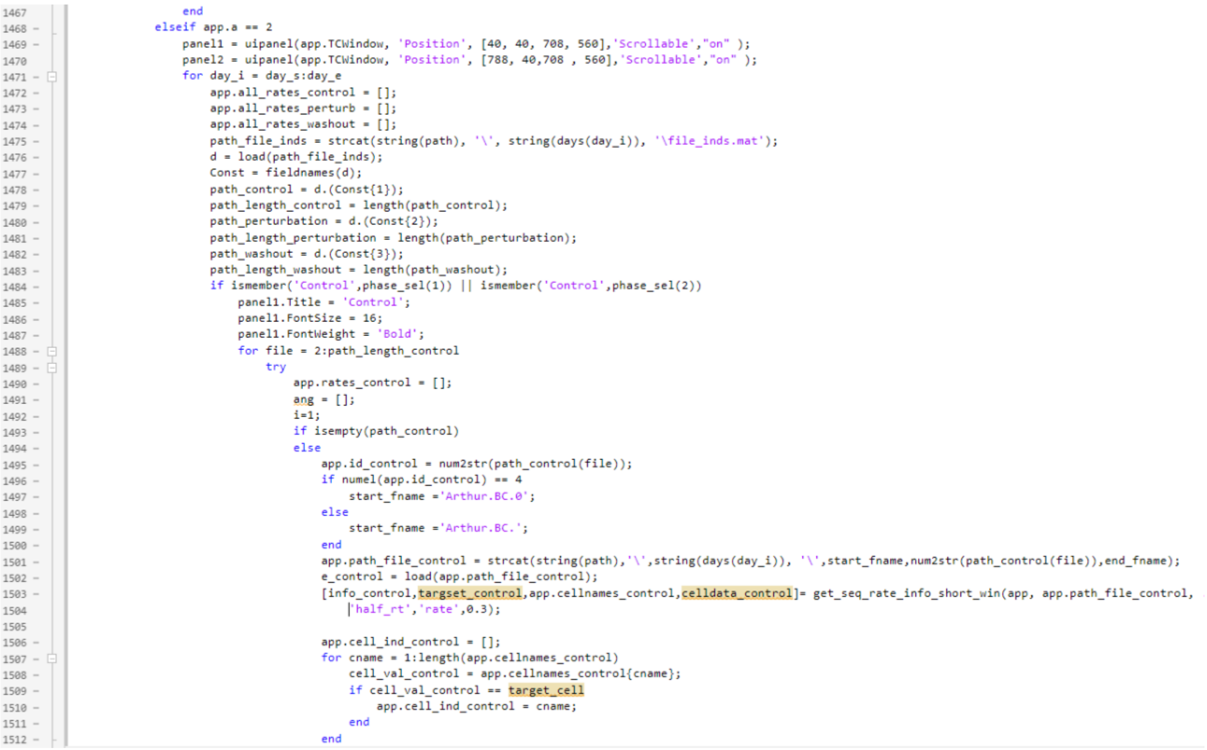
```
                    sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
                    sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
                    app.rates_perturb(:,i) = sr_cell_perturb';

                    early_perturb_fit = compute_cosine_fit(targets_perturb,app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
                end
            end
            app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
        catch err
            %errordlg(err.message)
        end
    end
    try
        mean_all_rates_perturb = mean(app.all_rates_perturb');
        [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
        angle_max_perturb = app.angle_perturb(index_max_value_perturb);
        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);
        app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
        ax_perturb = uiaxes(panel,'Position',[40, 140*(num_days-day_i)+10,1256,130]);
        hold(ax_perturb, 'on')
        plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
        plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
        plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
        xlabel(ax_perturb,'Angle','FontSize',12)
        ylabel(ax_perturb, 'Spiking rate','FontSize',12)
        title(ax_perturb, strcat('Day: ', days(day_i)))
        hold(ax_perturb,"off")
    catch err
        %errordlg(err.message)
    end
else
    app.ChooseDaysListBox.Items = string(phase_sel);
    panel.Title = 'Washout';
    panel.FontSize = 16;
    panel.FontWeight = 'Bold';
    for file = 2:path_length_washout
        try
            app.rates_washout = [];
            ang = [];
            i=1;
            if isempty(path_washout)
            else
```



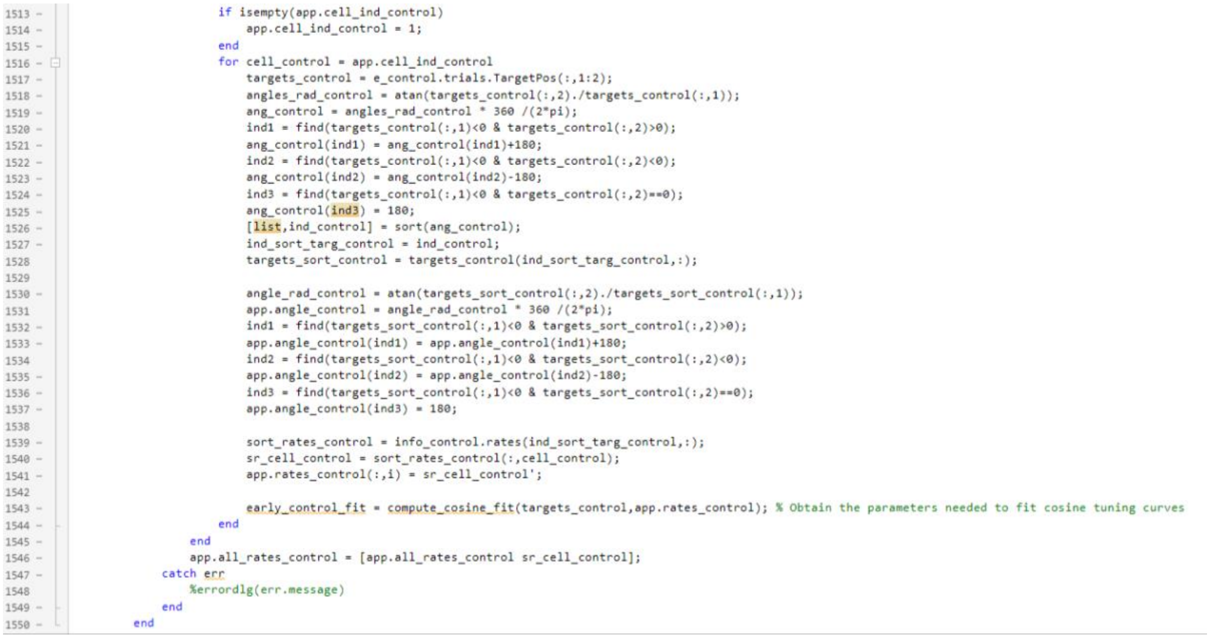
```
                app.id_washout = num2str(path_washout(file));
                if numel(app.id_washout) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
                app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
                e_washout = load(app.path_file_washout);
                [info_washout,targset_washout,app.cellnames_washout,celldata_washout]= get_seq_rate_info_short_win(app, app.path_file_washout,'half_rt','rate',0.3);

                app.cell_ind_washout = [];
                for cname = 1:length(app.cellnames_washout)
                    cell_val_washout = app.cellnames_washout{cname};
                    if cell_val_washout == target_cell
                        app.cell_ind_washout = cname;
                    end
                end
                if isempty(app.cell_ind_washout)
                    app.cell_ind_washout = 1;
                end
                for cell_washout = app.cell_ind_washout
                    targets_washout = e_washout.trials.TargetPos(:,1:2);
                    angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
                    ang_washout = angles_rad_washout * 360 /(2*pi);
                    ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
                    ang_washout(ind1) = ang_washout(ind1)+180;
                    ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
                    ang_washout(ind2) = ang_washout(ind2)-180;
                    ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
                    ang_washout(ind3) = 180;
                    [list,ind_washout] = sort(ang_washout);
                    ind_sort_targ_washout = ind_washout;
                    targets_sort_washout = targets_washout(ind_sort_targ_washout,:);

                    angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
                    app.angle_washout = angle_rad_washout * 360 /(2*pi);
                    ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
                    app.angle_washout(ind1) = app.angle_washout(ind1)+180;
                    ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
                    app.angle_washout(ind2) = app.angle_washout(ind2)-180;
                    ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
                    app.angle_washout(ind3) = 180;
```



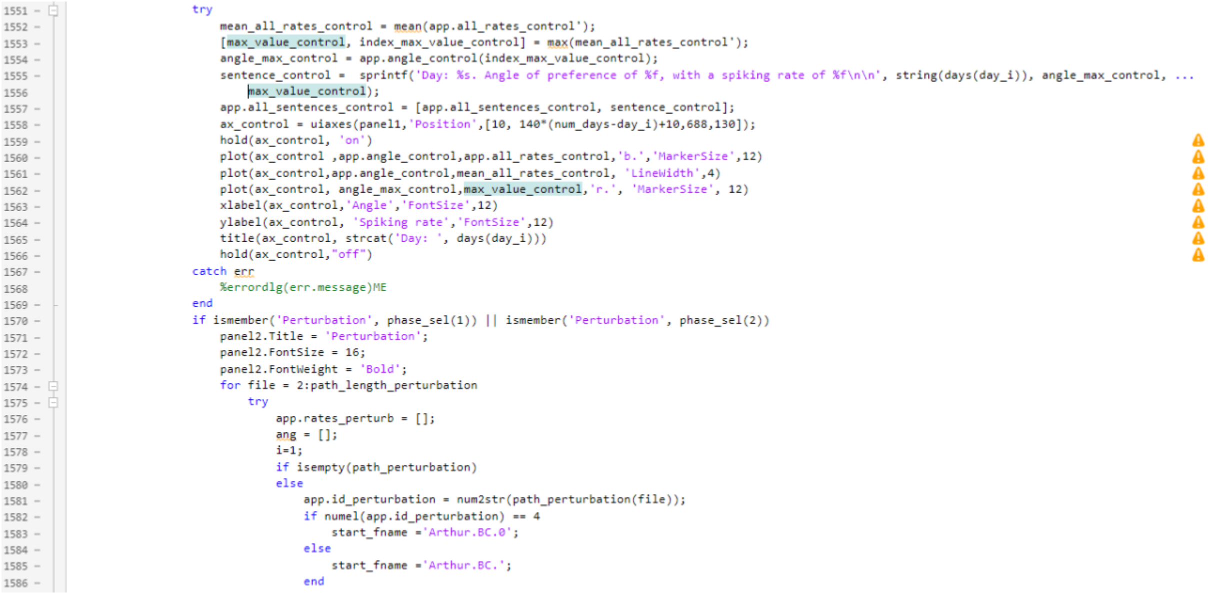
```

                    sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
                    sr_cell_washout = sort_rates_washout(:,cell_washout);
                    app.rates_washout(:,i) = sr_cell_washout';

                    early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
                end
            end
            app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
        catch err
            %errordlg(err.message)
        end
    end
    try
        mean_all_rates_washout = mean(app.all_rates_washout');
        [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
        angle_max_washout = app.angle_washout(index_max_value_washout);
        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);
        app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
        ax_washout = uiaxes(panel,'Position',[40, 140*(num_days-day_i)+10,1256,130]);
        hold(ax_washout, 'on')
        plot(ax_washout ,app.angle_washout,app.all_rates_washout,'b.','MarkerSize',12)
        plot(ax_washout,app.angle_washout,mean_all_rates_washout, 'LineWidth',4)
        plot(ax_washout, angle_max_washout,max_value_washout,'r.', 'MarkerSize', 12)
        xlabel(ax_washout,'Angle','FontSize',12)
        ylabel(ax_washout, 'Spiking rate','FontSize',12)
        title(ax_washout, strcat('Day: ', days(day_i)))
        hold(ax_washout,"off")
    catch err
        %errordlg(err.message)
    end
end
```



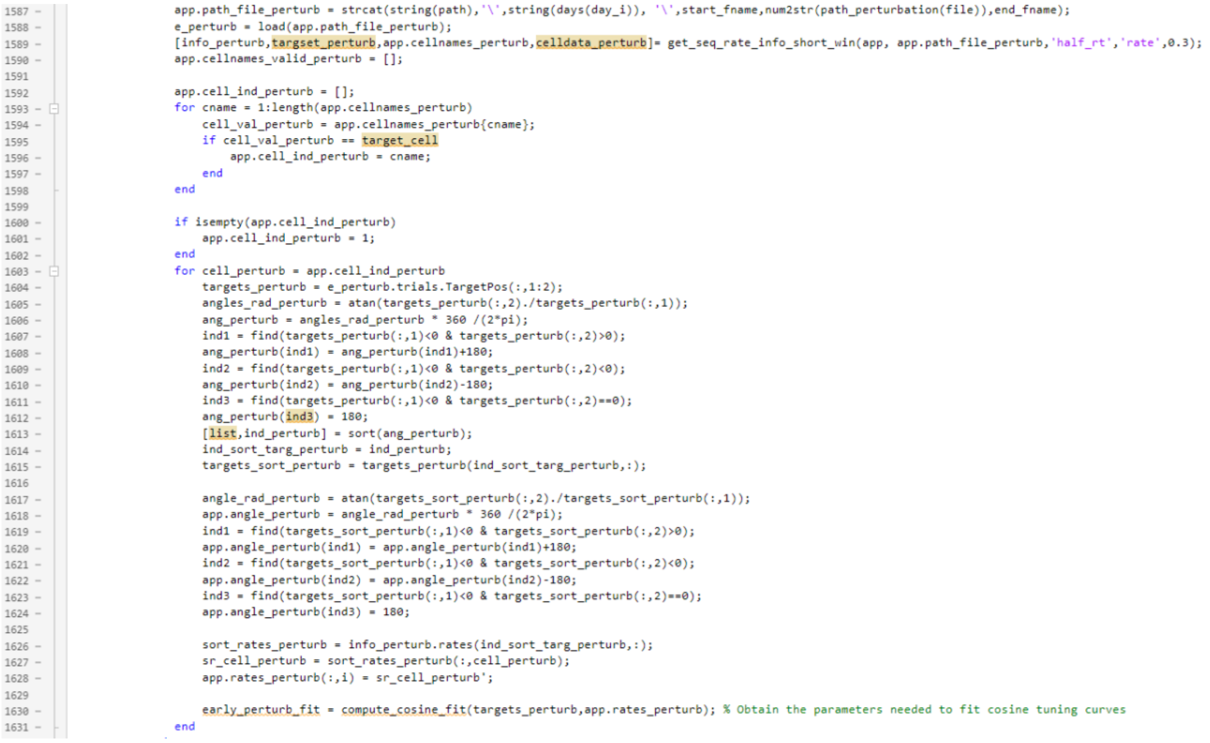
```
        end
    elseif app.a == 2
        panel1 = uipanel(app.TCWindow, 'Position', [40, 40, 708, 560],'Scrollable',"on" );
        panel2 = uipanel(app.TCWindow, 'Position', [788, 40,708 , 560],'Scrollable',"on" );
        for day_i = day_s:day_e
            app.all_rates_control = [];
            app.all_rates_perturb = [];
            app.all_rates_washout = [];
            path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
            d = load(path_file_inds);
            Const = fieldnames(d);
            path_control = d.(Const{1});
            path_length_control = length(path_control);
            path_perturbation = d.(Const{2});
            path_length_perturbation = length(path_perturbation);
            path_washout = d.(Const{3});
            path_length_washout = length(path_washout);
            if ismember('Control',phase_sel(1)) || ismember('Control',phase_sel(2))
                panel1.Title = 'Control';
                panel1.FontSize = 16;
                panel1.FontWeight = 'Bold';
                for file = 2:path_length_control
                    try
                        app.rates_control = [];
                        ang = [];
                        i=1;
                        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
                            app.path_file_control = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_control(file)),end_fname);
                            e_control = load(app.path_file_control);
                            [info_control,targset_control,app.cellnames_control,celldata_control]= get_seq_rate_info_short_win(app, app.path_file_control, 
                                'half_rt','rate',0.3);

                            app.cell_ind_control = [];
                            for cname = 1:length(app.cellnames_control)
                                cell_val_control = app.cellnames_control{cname};
                                if cell_val_control == target_cell
                                    app.cell_ind_control = cname;
                                end
                            end
```



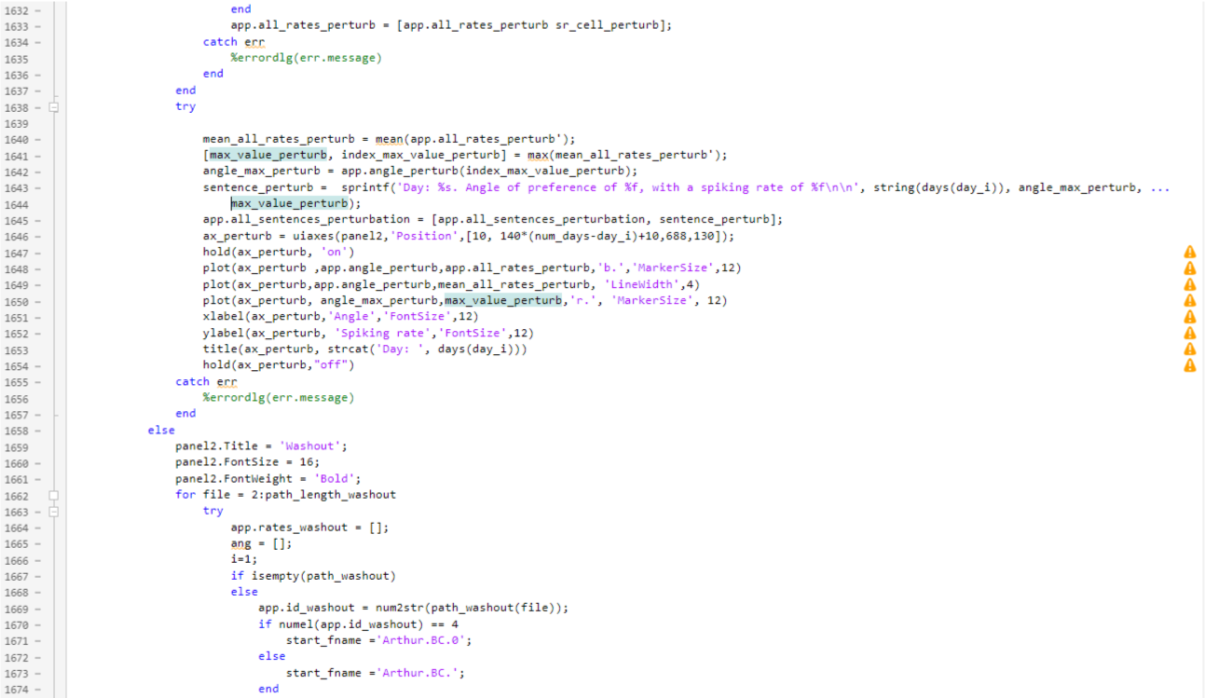
```
            if isempty(app.cell_ind_control)
                app.cell_ind_control = 1;
            end
            for cell_control = app.cell_ind_control
                targets_control = e_control.trials.TargetPos(:,1:2);
                angles_rad_control = atan(targets_control(:,2)./targets_control(:,1));
                ang_control = angles_rad_control * 360 /(2*pi);
                ind1 = find(targets_control(:,1)<0 & targets_control(:,2)>0);
                ang_control(ind1) = ang_control(ind1)+180;
                ind2 = find(targets_control(:,1)<0 & targets_control(:,2)<0);
                ang_control(ind2) = ang_control(ind2)-180;
                ind3 = find(targets_control(:,1)<0 & targets_control(:,2)==0);
                ang_control(ind3) = 180;
                [list,ind_control] = sort(ang_control);
                ind_sort_targ_control = ind_control;
                targets_sort_control = targets_control(ind_sort_targ_control,:);

                angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
                app.angle_control = angle_rad_control * 360 /(2*pi);
                ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
                app.angle_control(ind1) = app.angle_control(ind1)+180;
                ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
                app.angle_control(ind2) = app.angle_control(ind2)-180;
                ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
                app.angle_control(ind3) = 180;

                sort_rates_control = info_control.rates(ind_sort_targ_control,:);
                sr_cell_control = sort_rates_control(:,cell_control);
                app.rates_control(:,i) = sr_cell_control';

                early_control_fit = compute_cosine_fit(targets_control,app.rates_control); % Obtain the parameters needed to fit cosine tuning curves
            end
        end
        app.all_rates_control = [app.all_rates_control sr_cell_control];
    catch err
        %errordlg(err.message)
    end
end
```



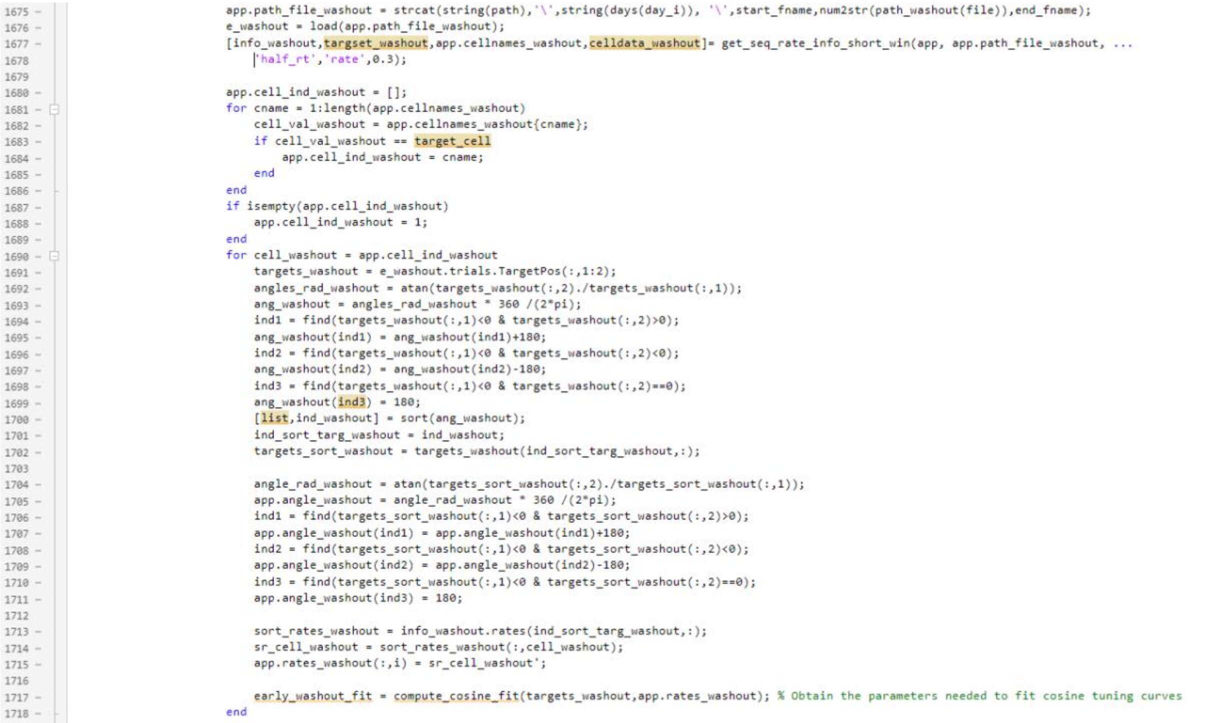
```
try
    mean_all_rates_control = mean(app.all_rates_control');
    [max_value_control, index_max_value_control] = max(mean_all_rates_control');
    angle_max_control = app.angle_control(index_max_value_control);
    sentence_control =  sprintf('Day: %s. Angle of preference of %f, with a spiking rate of %f\n\n', string(days(day_i)), angle_max_control, ...
        max_value_control);
    app.all_sentences_control = [app.all_sentences_control, sentence_control];
    ax_control = uiaxes(panel1,'Position',[10, 140*(num_days-day_i)+10,688,130]);
    hold(ax_control, 'on')
    plot(ax_control ,app.angle_control,app.all_rates_control,'b.','MarkerSize',12)
    plot(ax_control,app.angle_control,mean_all_rates_control, 'LineWidth',4)
    plot(ax_control, angle_max_control,max_value_control,'r.', 'MarkerSize', 12)
    xlabel(ax_control,'Angle','FontSize',12)
    ylabel(ax_control, 'Spiking rate','FontSize',12)
    title(ax_control, strcat('Day: ', days(day_i)))
    hold(ax_control,"off")
catch err
    %errordlg(err.message)ME
end
if ismember('Perturbation', phase_sel(1)) || ismember('Perturbation', phase_sel(2))
    panel2.Title = 'Perturbation';
    panel2.FontSize = 16;
    panel2.FontWeight = 'Bold';
    for file = 2:path_length_perturbation
        try
            app.rates_perturb = [];
            ang = [];
            i=1;
            if isempty(path_perturbation)
            else
                app.id_perturbation = num2str(path_perturbation(file));
                if numel(app.id_perturbation) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
```



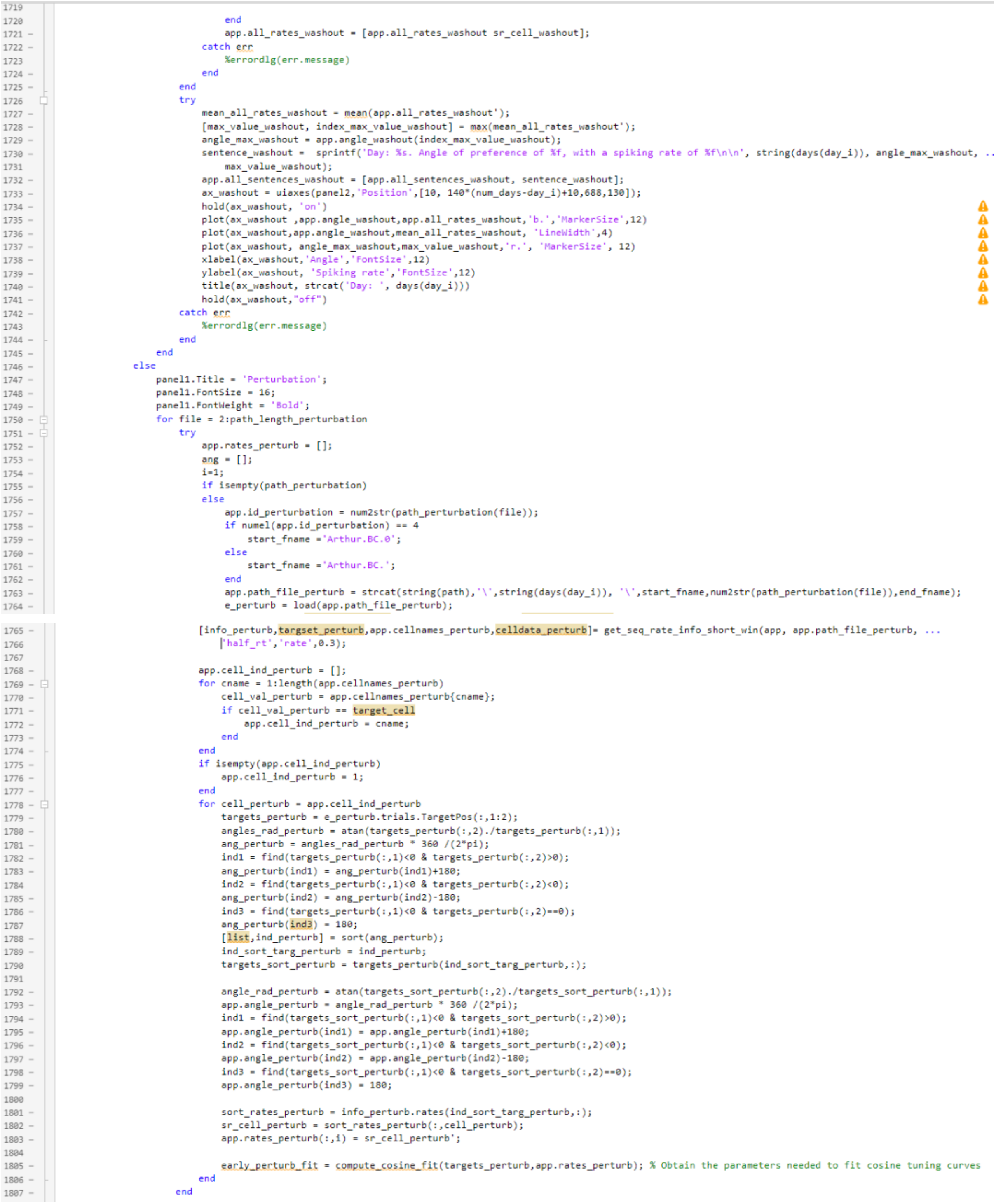
```
app.path_file_perturb = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
e_perturb = load(app.path_file_perturb);
[info_perturb,targset_perturb,app.cellnames_perturb,celldata_perturb]= get_seq_rate_info_short_win(app, app.path_file_perturb,'half_rt','rate',0.3);
app.cellnames_valid_perturb = [];

app.cell_ind_perturb = [];
for cname = 1:length(app.cellnames_perturb)
    cell_val_perturb = app.cellnames_perturb{cname};
    if cell_val_perturb == target_cell
        app.cell_ind_perturb = cname;
    end
end

if isempty(app.cell_ind_perturb)
    app.cell_ind_perturb = 1;
end
for cell_perturb = app.cell_ind_perturb
    targets_perturb = e_perturb.trials.TargetPos(:,1:2);
    angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
    ang_perturb = angles_rad_perturb * 360 /(2*pi);
    ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
    ang_perturb(ind1) = ang_perturb(ind1)+180;
    ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
    ang_perturb(ind2) = ang_perturb(ind2)-180;
    ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
    ang_perturb(ind3) = 180;
    [list,ind_perturb] = sort(ang_perturb);
    ind_sort_targ_perturb = ind_perturb;
    targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);

    angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
    app.angle_perturb = angle_rad_perturb * 360 /(2*pi);
    ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
    app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
    ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
    app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
    ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
    app.angle_perturb(ind3) = 180;

    sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
    sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
    app.rates_perturb(:,i) = sr_cell_perturb';

    early_perturb_fit = compute_cosine_fit(targets_perturb,app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
end
```



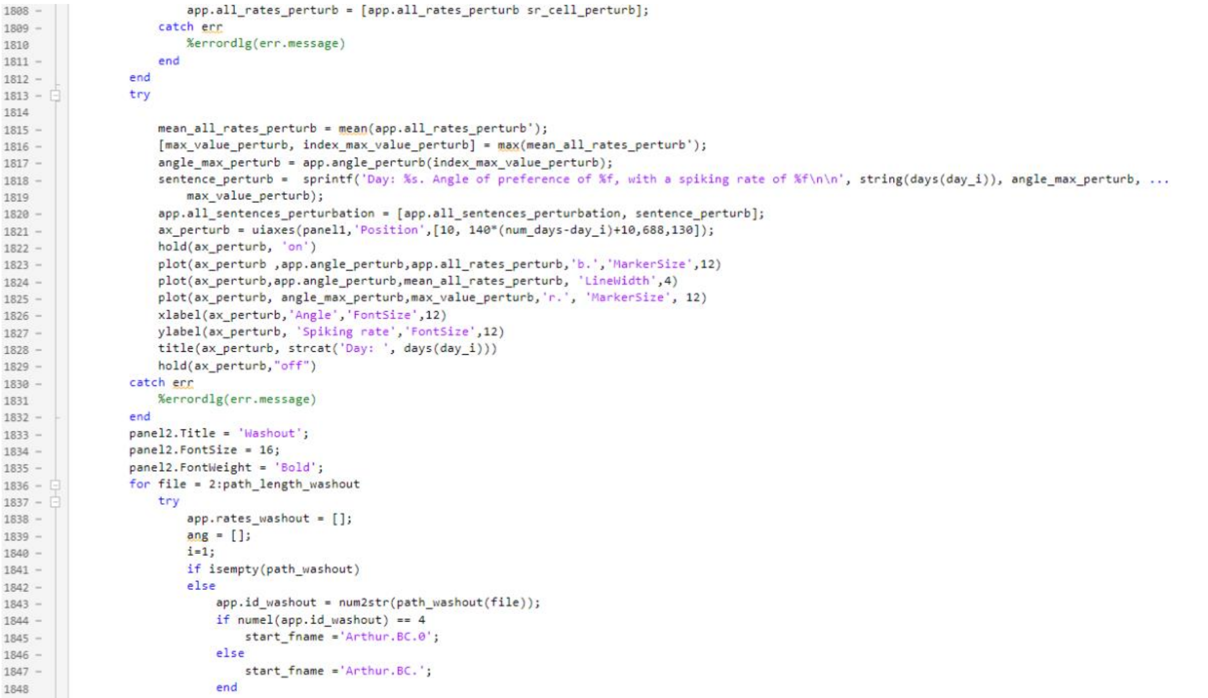
```
            end
            app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
        catch err
            %errordlg(err.message)
        end
    end
    try

        mean_all_rates_perturb = mean(app.all_rates_perturb');
        [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
        angle_max_perturb = app.angle_perturb(index_max_value_perturb);
        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);
        app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
        ax_perturb = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,688,130]);
        hold(ax_perturb, 'on')
        plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
        plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
        plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
        xlabel(ax_perturb,'Angle','FontSize',12)
        ylabel(ax_perturb, 'Spiking rate','FontSize',12)
        title(ax_perturb, strcat('Day: ', days(day_i)))
        hold(ax_perturb,"off")
    catch err
        %errordlg(err.message)
    end
else
    panel2.Title = 'Washout';
    panel2.FontSize = 16;
    panel2.FontWeight = 'Bold';
    for file = 2:path_length_washout
        try
            app.rates_washout = [];
            ang = [];
            i=1;
            if isempty(path_washout)
            else
                app.id_washout = num2str(path_washout(file));
                if numel(app.id_washout) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
```



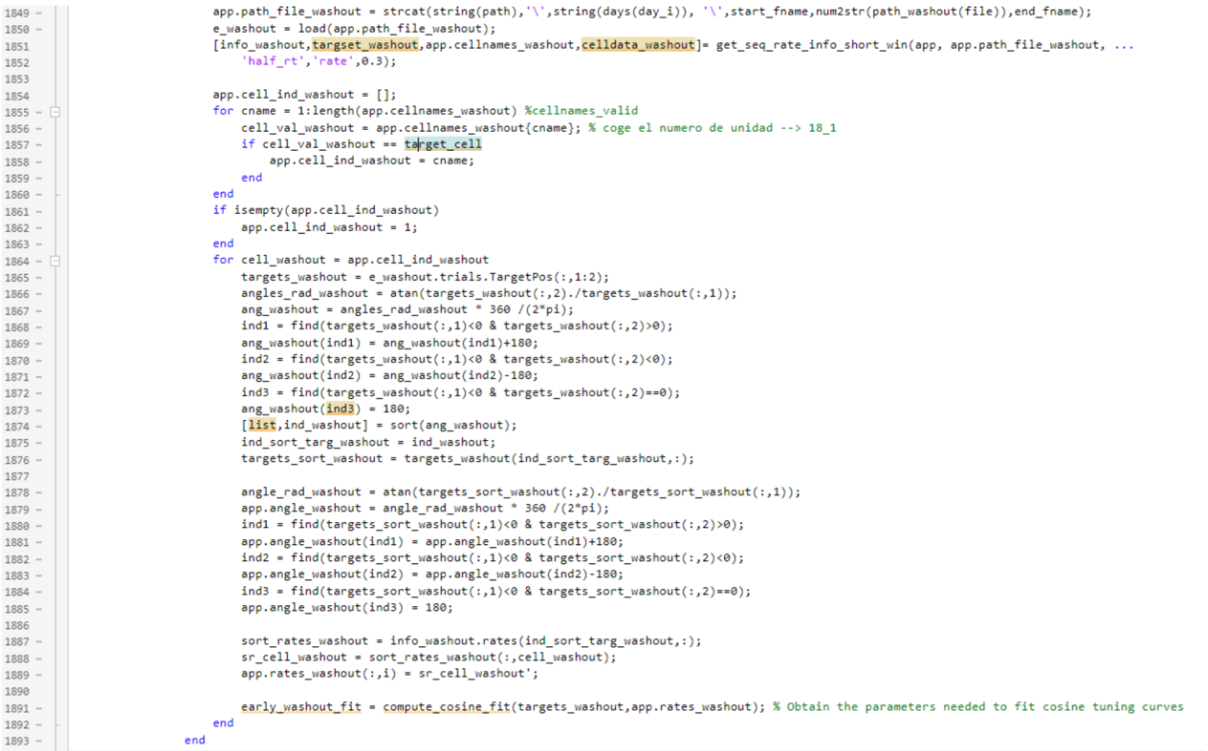
```
            app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
            e_washout = load(app.path_file_washout);
            [info_washout,targset_washout,app.cellnames_washout,celldata_washout]= get_seq_rate_info_short_win(app, app.path_file_washout, ...
                'half_rt','rate',0.3);

            app.cell_ind_washout = [];
            for cname = 1:length(app.cellnames_washout)
                cell_val_washout = app.cellnames_washout{cname};
                if cell_val_washout == target_cell
                    app.cell_ind_washout = cname;
                end
            end
            if isempty(app.cell_ind_washout)
                app.cell_ind_washout = 1;
            end
            for cell_washout = app.cell_ind_washout
                targets_washout = e_washout.trials.TargetPos(:,1:2);
                angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
                ang_washout = angles_rad_washout * 360 /(2*pi);
                ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
                ang_washout(ind1) = ang_washout(ind1)+180;
                ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
                ang_washout(ind2) = ang_washout(ind2)-180;
                ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
                ang_washout(ind3) = 180;
                [list,ind_washout] = sort(ang_washout);
                ind_sort_targ_washout = ind_washout;
                targets_sort_washout = targets_washout(ind_sort_targ_washout,:);

                angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
                app.angle_washout = angle_rad_washout * 360 /(2*pi);
                ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
                app.angle_washout(ind1) = app.angle_washout(ind1)+180;
                ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
                app.angle_washout(ind2) = app.angle_washout(ind2)-180;
                ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
                app.angle_washout(ind3) = 180;

                sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
                sr_cell_washout = sort_rates_washout(:,cell_washout);
                app.rates_washout(:,i) = sr_cell_washout';

                early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
            end
```



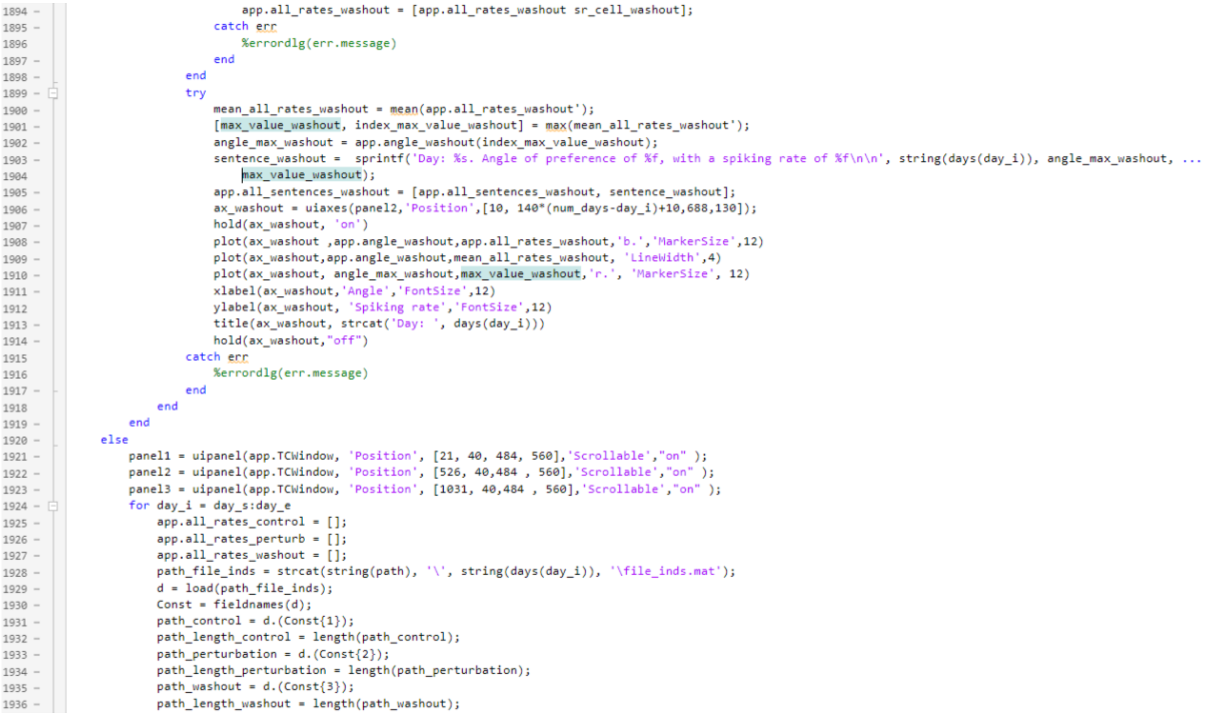
```
                end
                app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
            catch err
                %errordlg(err.message)
            end
        end
        try
            mean_all_rates_washout = mean(app.all_rates_washout');
            [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
            angle_max_washout = app.angle_washout(index_max_value_washout);
            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);
            app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
            ax_washout = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,688,130]);
            hold(ax_washout, 'on')
            plot(ax_washout ,app.angle_washout,app.all_rates_washout,'b.','MarkerSize',12)
            plot(ax_washout,app.angle_washout,mean_all_rates_washout, 'LineWidth',4)
            plot(ax_washout, angle_max_washout,max_value_washout,'r.', 'MarkerSize', 12)
            xlabel(ax_washout,'Angle','FontSize',12)
            ylabel(ax_washout, 'Spiking rate','FontSize',12)
            title(ax_washout, strcat('Day: ', days(day_i)))
            hold(ax_washout,"off")
        catch err
            %errordlg(err.message)
        end
    end
else
    panel1.Title = 'Perturbation';
    panel1.FontSize = 16;
    panel1.FontWeight = 'Bold';
    for file = 2:path_length_perturbation
        try
            app.rates_perturb = [];
            ang = [];
            i=1;
            if isempty(path_perturbation)
            else
                app.id_perturbation = num2str(path_perturbation(file));
                if numel(app.id_perturbation) == 4
                    start_fname ='Arthur.BC.0';
                else
                    start_fname ='Arthur.BC.';
                end
                app.path_file_perturb = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
                e_perturb = load(app.path_file_perturb);

            [info_perturb,targset_perturb,app.cellnames_perturb,celldata_perturb]= get_seq_rate_info_short_win(app, app.path_file_perturb, ...
                'half_rt','rate',0.3);

            app.cell_ind_perturb = [];
            for cname = 1:length(app.cellnames_perturb)
                cell_val_perturb = app.cellnames_perturb{cname};
                if cell_val_perturb == target_cell
                    app.cell_ind_perturb = cname;
                end
            end
            if isempty(app.cell_ind_perturb)
                app.cell_ind_perturb = 1;
            end
            for cell_perturb = app.cell_ind_perturb
                targets_perturb = e_perturb.trials.TargetPos(:,1:2);
                angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
                ang_perturb = angles_rad_perturb * 360 /(2*pi);
                ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
                ang_perturb(ind1) = ang_perturb(ind1)+180;
                ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
                ang_perturb(ind2) = ang_perturb(ind2)-180;
                ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
                ang_perturb(ind3) = 180;
                [list,ind_perturb] = sort(ang_perturb);
                ind_sort_targ_perturb = ind_perturb;
                targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);

                angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
                app.angle_perturb = angle_rad_perturb * 360 /(2*pi);
                ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
                app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
                ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
                app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
                ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
                app.angle_perturb(ind3) = 180;

                sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
                sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
                app.rates_perturb(:,i) = sr_cell_perturb';

                early_perturb_fit = compute_cosine_fit(targets_perturb,app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
            end
        end
```



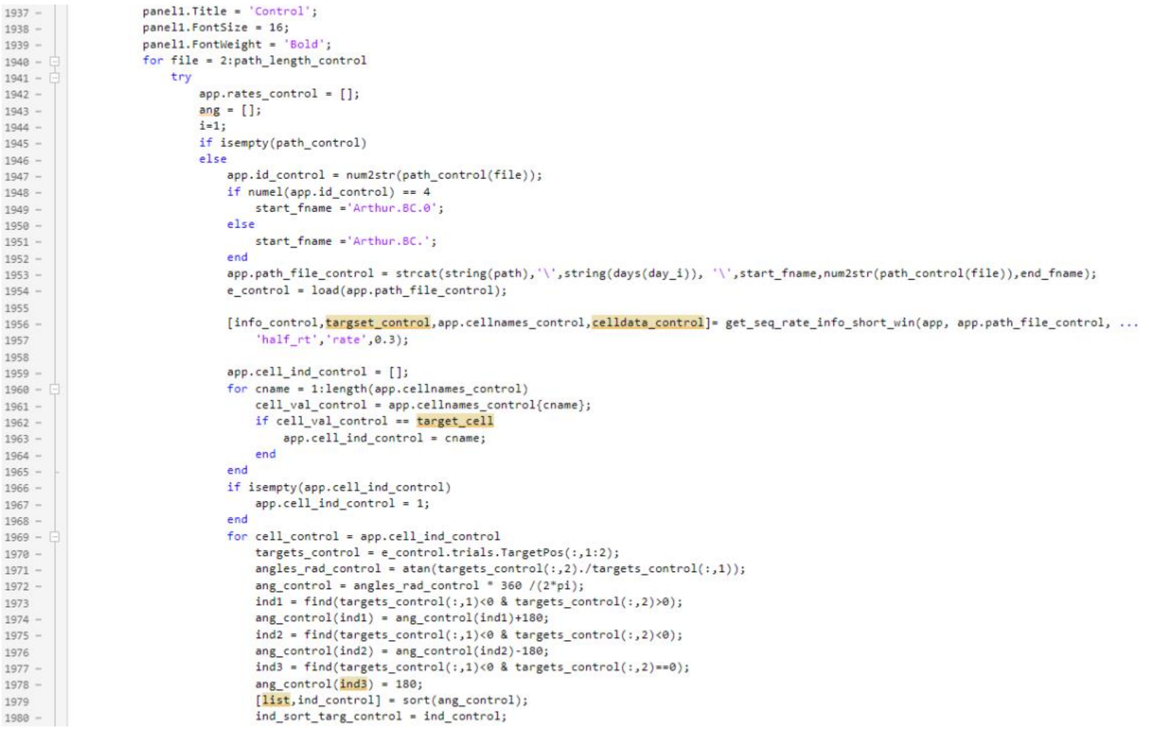
```
                app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
            catch err
                %errordlg(err.message)
            end
        end
        try

            mean_all_rates_perturb = mean(app.all_rates_perturb');
            [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
            angle_max_perturb = app.angle_perturb(index_max_value_perturb);
            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);
            app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
            ax_perturb = uiaxes(panel1,'Position',[10, 140*(num_days-day_i)+10,688,130]);
            hold(ax_perturb, 'on')
            plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
            plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
            plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
            xlabel(ax_perturb,'Angle','FontSize',12)
            ylabel(ax_perturb, 'Spiking rate','FontSize',12)
            title(ax_perturb, strcat('Day: ', days(day_i)))
            hold(ax_perturb,"off")
        catch err
            %errordlg(err.message)
        end
        panel2.Title = 'Washout';
        panel2.FontSize = 16;
        panel2.FontWeight = 'Bold';
        for file = 2:path_length_washout
            try
                app.rates_washout = [];
                ang = [];
                i=1;
                if isempty(path_washout)
                else
                    app.id_washout = num2str(path_washout(file));
                    if numel(app.id_washout) == 4
                        start_fname ='Arthur.BC.0';
                    else
                        start_fname ='Arthur.BC.';
                    end
```



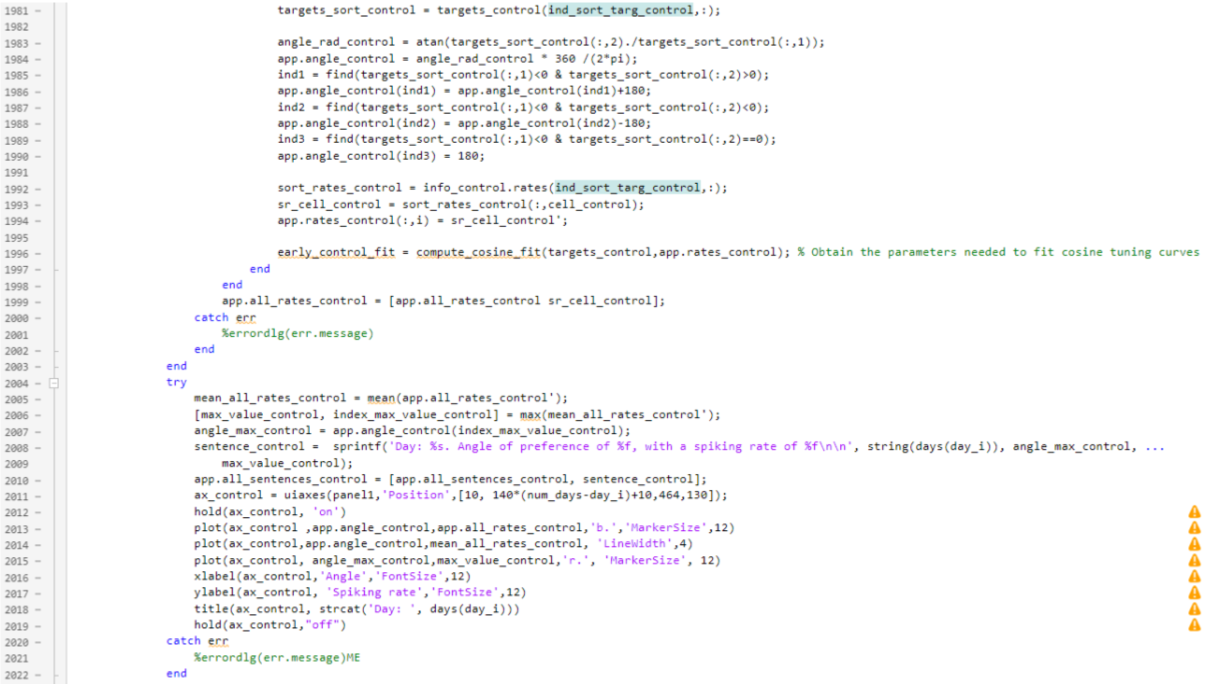
```
                    app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
                    e_washout = load(app.path_file_washout);
                    [info_washout,targset_washout,app.cellnames_washout,celldata_washout]= get_seq_rate_info_short_win(app, app.path_file_washout, ...
                        'half_rt','rate',0.3);

                    app.cell_ind_washout = [];
                    for cname = 1:length(app.cellnames_washout) %cellnames_valid
                        cell_val_washout = app.cellnames_washout{cname}; % coge el numero de unidad --> 18_1
                        if cell_val_washout == target_cell
                            app.cell_ind_washout = cname;
                        end
                    end
                    if isempty(app.cell_ind_washout)
                        app.cell_ind_washout = 1;
                    end
                    for cell_washout = app.cell_ind_washout
                        targets_washout = e_washout.trials.TargetPos(:,1:2);
                        angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
                        ang_washout = angles_rad_washout * 360 /(2*pi);
                        ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
                        ang_washout(ind1) = ang_washout(ind1)+180;
                        ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
                        ang_washout(ind2) = ang_washout(ind2)-180;
                        ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
                        ang_washout(ind3) = 180;
                        [list,ind_washout] = sort(ang_washout);
                        ind_sort_targ_washout = ind_washout;
                        targets_sort_washout = targets_washout(ind_sort_targ_washout,:);

                        angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
                        app.angle_washout = angle_rad_washout * 360 /(2*pi);
                        ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
                        app.angle_washout(ind1) = app.angle_washout(ind1)+180;
                        ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
                        app.angle_washout(ind2) = app.angle_washout(ind2)-180;
                        ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
                        app.angle_washout(ind3) = 180;

                        sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
                        sr_cell_washout = sort_rates_washout(:,cell_washout);
                        app.rates_washout(:,i) = sr_cell_washout';

                        early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
                    end
                end
```



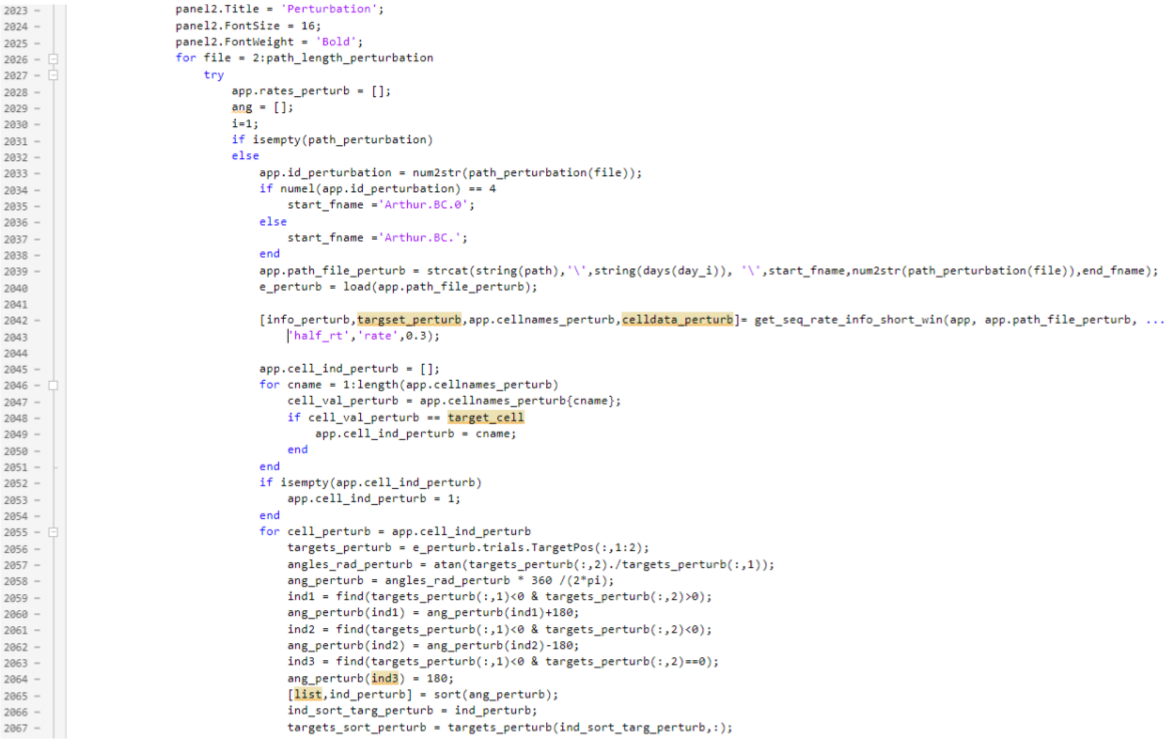
```
                        app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
                    catch err
                        %errordlg(err.message)
                    end
                end
                try
                    mean_all_rates_washout = mean(app.all_rates_washout');
                    [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
                    angle_max_washout = app.angle_washout(index_max_value_washout);
                    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);
                    app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
                    ax_washout = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,688,130]);
                    hold(ax_washout, 'on')
                    plot(ax_washout ,app.angle_washout,app.all_rates_washout,'b.','MarkerSize',12)
                    plot(ax_washout,app.angle_washout,mean_all_rates_washout, 'LineWidth',4)
                    plot(ax_washout, angle_max_washout,max_value_washout,'r.', 'MarkerSize', 12)
                    xlabel(ax_washout,'Angle','FontSize',12)
                    ylabel(ax_washout, 'Spiking rate','FontSize',12)
                    title(ax_washout, strcat('Day: ', days(day_i)))
                    hold(ax_washout,"off")
                catch err
                    %errordlg(err.message)
                end
            end
        end
    else
        panel1 = uipanel(app.TCWindow, 'Position', [21, 40, 484, 560],'Scrollable',"on" );
        panel2 = uipanel(app.TCWindow, 'Position', [526, 40,484 , 560],'Scrollable',"on" );
        panel3 = uipanel(app.TCWindow, 'Position', [1031, 40,484 , 560],'Scrollable',"on" );
        for day_i = day_s:day_e
            app.all_rates_control = [];
            app.all_rates_perturb = [];
            app.all_rates_washout = [];
            path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
            d = load(path_file_inds);
            Const = fieldnames(d);
            path_control = d.(Const{1});
            path_length_control = length(path_control);
            path_perturbation = d.(Const{2});
            path_length_perturbation = length(path_perturbation);
            path_washout = d.(Const{3});
            path_length_washout = length(path_washout);
```



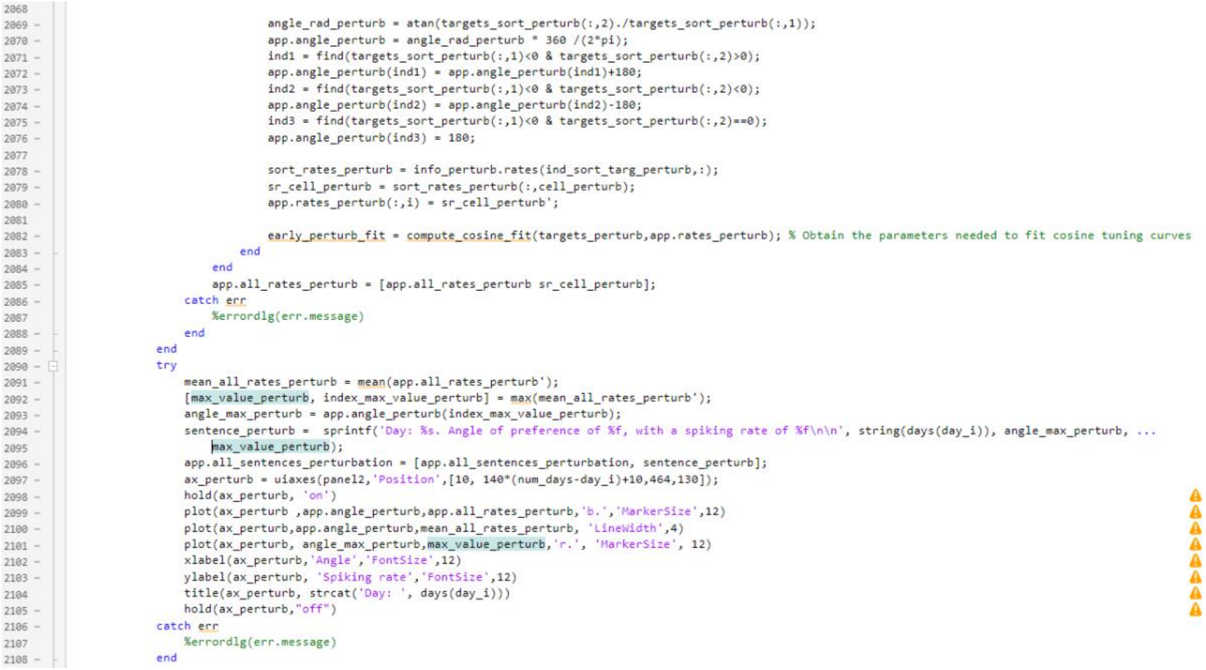
```
        panel1.Title = 'Control';
        panel1.FontSize = 16;
        panel1.FontWeight = 'Bold';
        for file = 2:path_length_control
            try
                app.rates_control = [];
                ang = [];
                i=1;
                if isempty(path_control)
                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
                    app.path_file_control = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_control(file)),end_fname);
                    e_control = load(app.path_file_control);

                    [info_control,targset_control,app.cellnames_control,celldata_control]= get_seq_rate_info_short_win(app, app.path_file_control, ...
                        'half_rt','rate',0.3);

                    app.cell_ind_control = [];
                    for cname = 1:length(app.cellnames_control)
                        cell_val_control = app.cellnames_control{cname};
                        if cell_val_control == target_cell
                            app.cell_ind_control = cname;
                        end
                    end
                    if isempty(app.cell_ind_control)
                        app.cell_ind_control = 1;
                    end
                    for cell_control = app.cell_ind_control
                        targets_control = e_control.trials.TargetPos(:,1:2);
                        angles_rad_control = atan(targets_control(:,2)./targets_control(:,1));
                        ang_control = angles_rad_control * 360 /(2*pi);
                        ind1 = find(targets_control(:,1)<0 & targets_control(:,2)>0);
                        ang_control(ind1) = ang_control(ind1)+180;
                        ind2 = find(targets_control(:,1)<0 & targets_control(:,2)<0);
                        ang_control(ind2) = ang_control(ind2)-180;
                        ind3 = find(targets_control(:,1)<0 & targets_control(:,2)==0);
                        ang_control(ind3) = 180;
                        [list,ind_control] = sort(ang_control);
                        ind_sort_targ_control = ind_control;
```



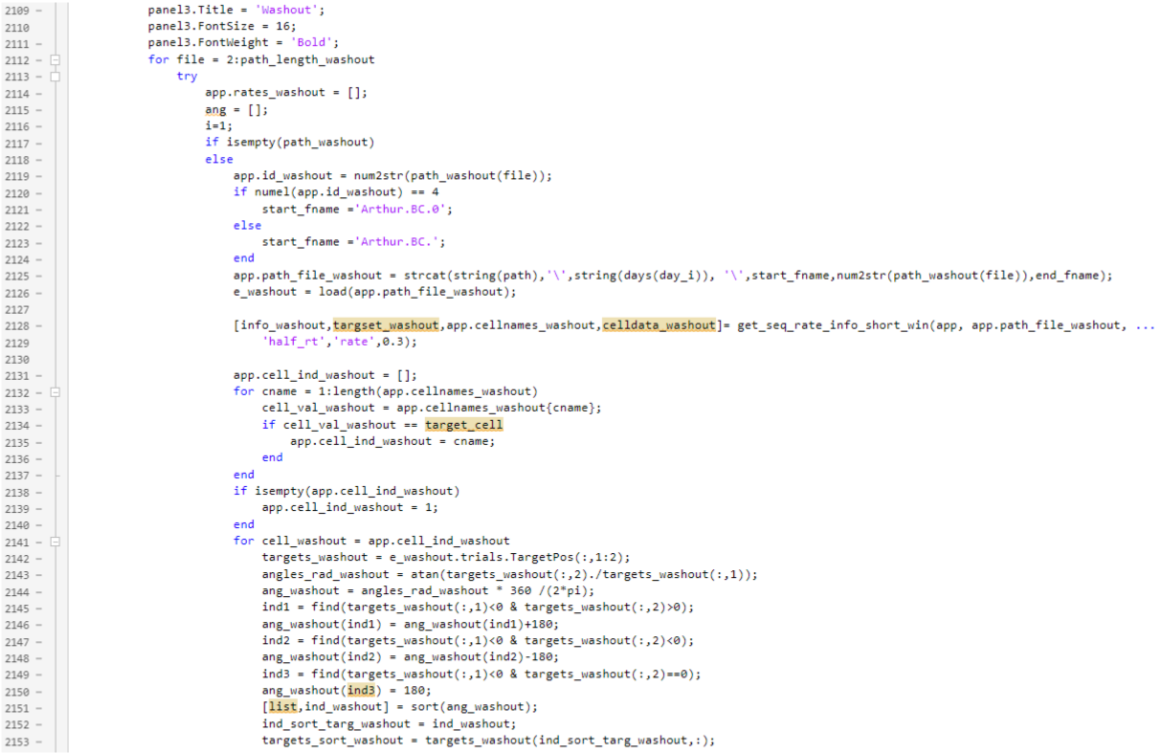
```
                targets_sort_control = targets_control(ind_sort_targ_control,:);

                angle_rad_control = atan(targets_sort_control(:,2)./targets_sort_control(:,1));
                app.angle_control = angle_rad_control * 360 /(2*pi);
                ind1 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)>0);
                app.angle_control(ind1) = app.angle_control(ind1)+180;
                ind2 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)<0);
                app.angle_control(ind2) = app.angle_control(ind2)-180;
                ind3 = find(targets_sort_control(:,1)<0 & targets_sort_control(:,2)==0);
                app.angle_control(ind3) = 180;

                sort_rates_control = info_control.rates(ind_sort_targ_control,:);
                sr_cell_control = sort_rates_control(:,cell_control);
                app.rates_control(:,i) = sr_cell_control';

                early_control_fit = compute_cosine_fit(targets_control,app.rates_control); % Obtain the parameters needed to fit cosine tuning curves
            end
        end
        app.all_rates_control = [app.all_rates_control sr_cell_control];
    catch err
        %errordlg(err.message)
    end
end
try
    mean_all_rates_control = mean(app.all_rates_control');
    [max_value_control, index_max_value_control] = max(mean_all_rates_control');
    angle_max_control = app.angle_control(index_max_value_control);
    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);
    app.all_sentences_control = [app.all_sentences_control, sentence_control];
    ax_control = uiaxes(panel1,'Position',[10, 140*(num_days-day_i)+10,464,130]);
    hold(ax_control, 'on')
    plot(ax_control ,app.angle_control,app.all_rates_control,'b.','MarkerSize',12)
    plot(ax_control,app.angle_control,mean_all_rates_control, 'LineWidth',4)
    plot(ax_control, angle_max_control,max_value_control,'r.', 'MarkerSize', 12)
    xlabel(ax_control,'Angle','FontSize',12)
    ylabel(ax_control, 'Spiking rate','FontSize',12)
    title(ax_control, strcat('Day: ', days(day_i)))
    hold(ax_control,"off")
catch err
    %errordlg(err.message)ME
end
```



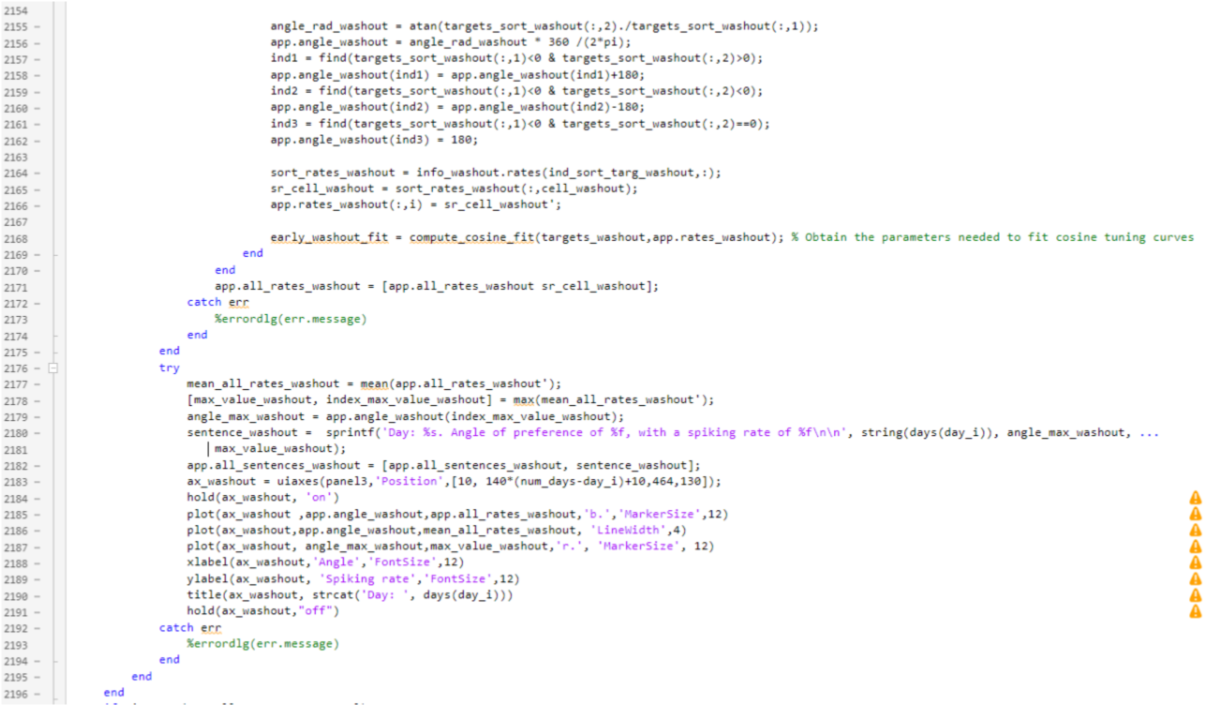
```
panel2.Title = 'Perturbation';
panel2.FontSize = 16;
panel2.FontWeight = 'Bold';
for file = 2:path_length_perturbation
    try
        app.rates_perturb = [];
        ang = [];
        i=1;
        if isempty(path_perturbation)
        else
            app.id_perturbation = num2str(path_perturbation(file));
            if numel(app.id_perturbation) == 4
                start_fname ='Arthur.BC.0';
            else
                start_fname ='Arthur.BC.';
            end
            app.path_file_perturb = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_perturbation(file)),end_fname);
            e_perturb = load(app.path_file_perturb);

            [info_perturb,targset_perturb,app.cellnames_perturb,celldata_perturb]= get_seq_rate_info_short_win(app, app.path_file_perturb, ...
                'half_rt','rate',0.3);

            app.cell_ind_perturb = [];
            for cname = 1:length(app.cellnames_perturb)
                cell_val_perturb = app.cellnames_perturb{cname};
                if cell_val_perturb == target_cell
                    app.cell_ind_perturb = cname;
                end
            end
            if isempty(app.cell_ind_perturb)
                app.cell_ind_perturb = 1;
            end
            for cell_perturb = app.cell_ind_perturb
                targets_perturb = e_perturb.trials.TargetPos(:,1:2);
                angles_rad_perturb = atan(targets_perturb(:,2)./targets_perturb(:,1));
                ang_perturb = angles_rad_perturb * 360 /(2*pi);
                ind1 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)>0);
                ang_perturb(ind1) = ang_perturb(ind1)+180;
                ind2 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)<0);
                ang_perturb(ind2) = ang_perturb(ind2)-180;
                ind3 = find(targets_perturb(:,1)<0 & targets_perturb(:,2)==0);
                ang_perturb(ind3) = 180;
                [list,ind_perturb] = sort(ang_perturb);
                ind_sort_targ_perturb = ind_perturb;
                targets_sort_perturb = targets_perturb(ind_sort_targ_perturb,:);
```



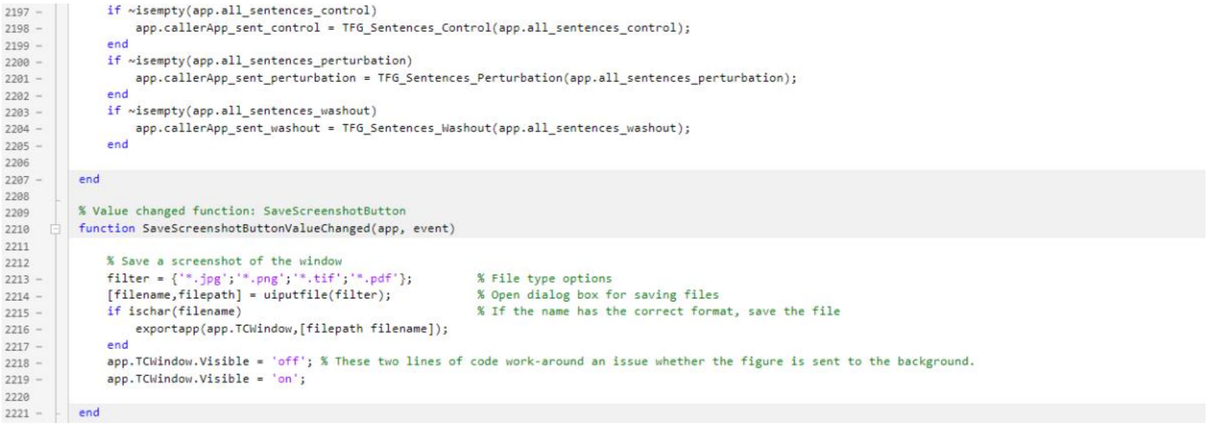
```
                angle_rad_perturb = atan(targets_sort_perturb(:,2)./targets_sort_perturb(:,1));
                app.angle_perturb = angle_rad_perturb * 360 /(2*pi);
                ind1 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)>0);
                app.angle_perturb(ind1) = app.angle_perturb(ind1)+180;
                ind2 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)<0);
                app.angle_perturb(ind2) = app.angle_perturb(ind2)-180;
                ind3 = find(targets_sort_perturb(:,1)<0 & targets_sort_perturb(:,2)==0);
                app.angle_perturb(ind3) = 180;

                sort_rates_perturb = info_perturb.rates(ind_sort_targ_perturb,:);
                sr_cell_perturb = sort_rates_perturb(:,cell_perturb);
                app.rates_perturb(:,i) = sr_cell_perturb';

                early_perturb_fit = compute_cosine_fit(targets_perturb,app.rates_perturb); % Obtain the parameters needed to fit cosine tuning curves
            end
        end
        app.all_rates_perturb = [app.all_rates_perturb sr_cell_perturb];
    catch err
        %errordlg(err.message)
    end
end
try
    mean_all_rates_perturb = mean(app.all_rates_perturb');
    [max_value_perturb, index_max_value_perturb] = max(mean_all_rates_perturb');
    angle_max_perturb = app.angle_perturb(index_max_value_perturb);
    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);
    app.all_sentences_perturbation = [app.all_sentences_perturbation, sentence_perturb];
    ax_perturb = uiaxes(panel2,'Position',[10, 140*(num_days-day_i)+10,464,130]);
    hold(ax_perturb, 'on')
    plot(ax_perturb ,app.angle_perturb,app.all_rates_perturb,'b.','MarkerSize',12)
    plot(ax_perturb,app.angle_perturb,mean_all_rates_perturb, 'LineWidth',4)
    plot(ax_perturb, angle_max_perturb,max_value_perturb,'r.', 'MarkerSize', 12)
    xlabel(ax_perturb,'Angle','FontSize',12)
    ylabel(ax_perturb, 'Spiking rate','FontSize',12)
    title(ax_perturb, strcat('Day: ', days(day_i)))
    hold(ax_perturb,"off")
catch err
    %errordlg(err.message)
end
```



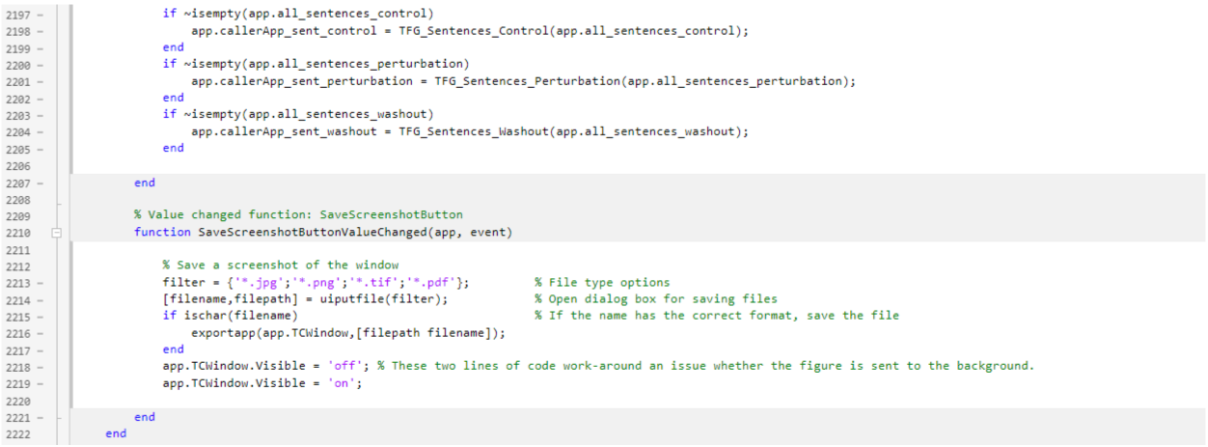
```
panel3.Title = 'Washout';
panel3.FontSize = 16;
panel3.FontWeight = 'Bold';
for file = 2:path_length_washout
    try
        app.rates_washout = [];
        ang = [];
        i=1;
        if isempty(path_washout)
        else
            app.id_washout = num2str(path_washout(file));
            if numel(app.id_washout) == 4
                start_fname ='Arthur.BC.0';
            else
                start_fname ='Arthur.BC.';
            end
            app.path_file_washout = strcat(string(path),'\',string(days(day_i)), '\',start_fname,num2str(path_washout(file)),end_fname);
            e_washout = load(app.path_file_washout);

            [info_washout,targset_washout,app.cellnames_washout,celldata_washout]= get_seq_rate_info_short_win(app, app.path_file_washout, ...
                'half_rt','rate',0.3);

            app.cell_ind_washout = [];
            for cname = 1:length(app.cellnames_washout)
                cell_val_washout = app.cellnames_washout{cname};
                if cell_val_washout == target_cell
                    app.cell_ind_washout = cname;
                end
            end
            if isempty(app.cell_ind_washout)
                app.cell_ind_washout = 1;
            end
            for cell_washout = app.cell_ind_washout
                targets_washout = e_washout.trials.TargetPos(:,1:2);
                angles_rad_washout = atan(targets_washout(:,2)./targets_washout(:,1));
                ang_washout = angles_rad_washout * 360 /(2*pi);
                ind1 = find(targets_washout(:,1)<0 & targets_washout(:,2)>0);
                ang_washout(ind1) = ang_washout(ind1)+180;
                ind2 = find(targets_washout(:,1)<0 & targets_washout(:,2)<0);
                ang_washout(ind2) = ang_washout(ind2)-180;
                ind3 = find(targets_washout(:,1)<0 & targets_washout(:,2)==0);
                ang_washout(ind3) = 180;
                [list,ind_washout] = sort(ang_washout);
                ind_sort_targ_washout = ind_washout;
                targets_sort_washout = targets_washout(ind_sort_targ_washout,:);
```



```

                        angle_rad_washout = atan(targets_sort_washout(:,2)./targets_sort_washout(:,1));
                        app.angle_washout = angle_rad_washout * 360 /(2*pi);
                        ind1 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)>0);
                        app.angle_washout(ind1) = app.angle_washout(ind1)+180;
                        ind2 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)<0);
                        app.angle_washout(ind2) = app.angle_washout(ind2)-180;
                        ind3 = find(targets_sort_washout(:,1)<0 & targets_sort_washout(:,2)==0);
                        app.angle_washout(ind3) = 180;

                        sort_rates_washout = info_washout.rates(ind_sort_targ_washout,:);
                        sr_cell_washout = sort_rates_washout(:,cell_washout);
                        app.rates_washout(:,i) = sr_cell_washout';

                        early_washout_fit = compute_cosine_fit(targets_washout,app.rates_washout); % Obtain the parameters needed to fit cosine tuning curves
                    end
                end
                app.all_rates_washout = [app.all_rates_washout sr_cell_washout];
            catch err
                %errordlg(err.message)
            end
        end
        try
            mean_all_rates_washout = mean(app.all_rates_washout');
            [max_value_washout, index_max_value_washout] = max(mean_all_rates_washout');
            angle_max_washout = app.angle_washout(index_max_value_washout);
            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);
            app.all_sentences_washout = [app.all_sentences_washout, sentence_washout];
            ax_washout = uiaxes(panel3,'Position',[10, 140*(num_days-day_i)+10,464,130]);
            hold(ax_washout, 'on')
            plot(ax_washout ,app.angle_washout,app.all_rates_washout,'b.','MarkerSize',12)
            plot(ax_washout,app.angle_washout,mean_all_rates_washout, 'LineWidth',4)
            plot(ax_washout, angle_max_washout,max_value_washout,'r.', 'MarkerSize', 12)
            xlabel(ax_washout,'Angle','FontSize',12)
            ylabel(ax_washout, 'Spiking rate','FontSize',12)
            title(ax_washout, strcat('Day: ', days(day_i)))
            hold(ax_washout,"off")
        catch err
            %errordlg(err.message)
        end
    end
end
```

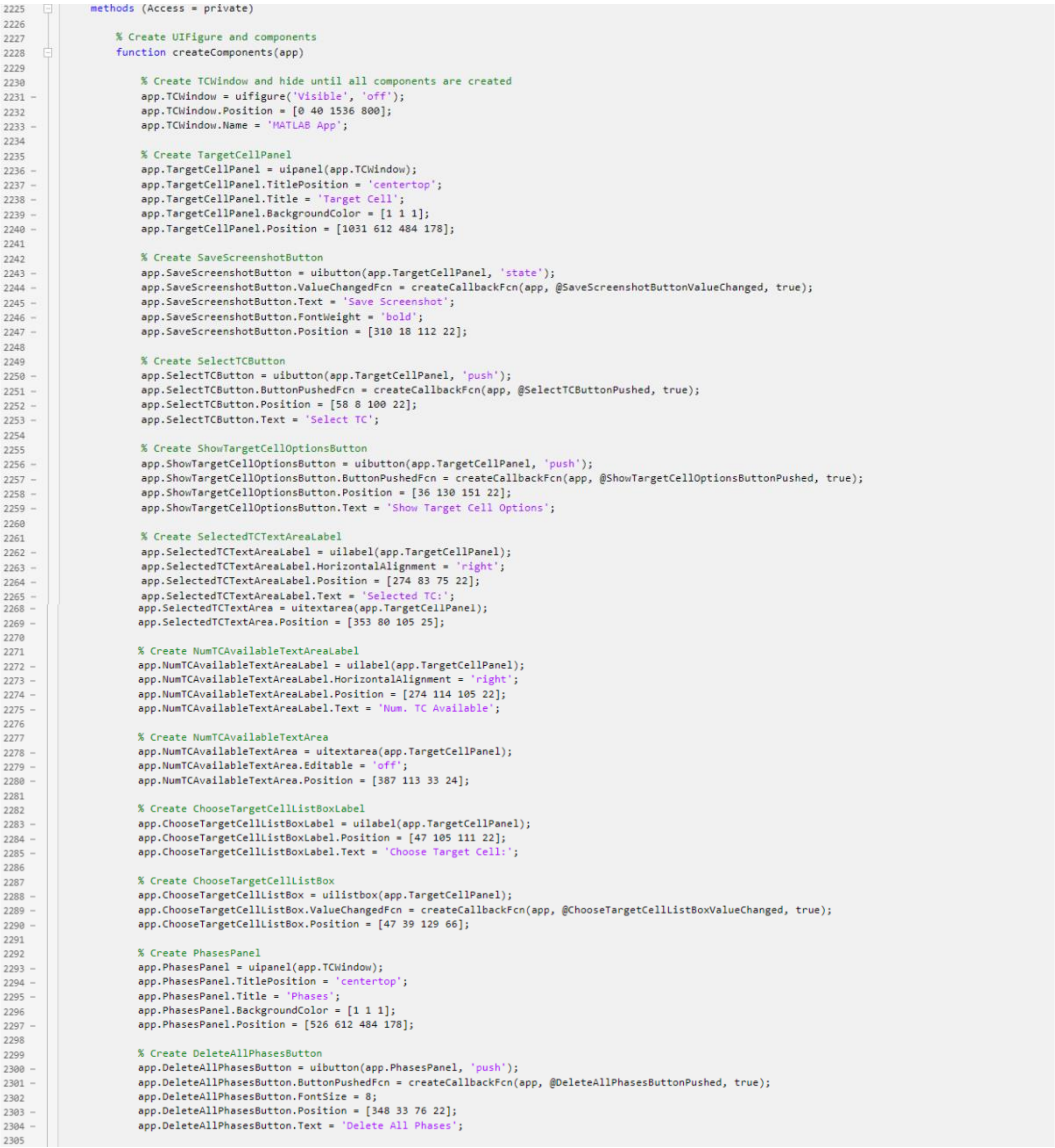


```
        if ~isempty(app.all_sentences_control)
            app.callerApp_sent_control = TFG_Sentences_Control(app.all_sentences_control);
        end
        if ~isempty(app.all_sentences_perturbation)
            app.callerApp_sent_perturbation = TFG_Sentences_Perturbation(app.all_sentences_perturbation);
        end
        if ~isempty(app.all_sentences_washout)
            app.callerApp_sent_washout = TFG_Sentences_Washout(app.all_sentences_washout);
        end

    end

    % Value changed function: SaveScreenshotButton
    function SaveScreenshotButtonValueChanged(app, event)

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

    end
```

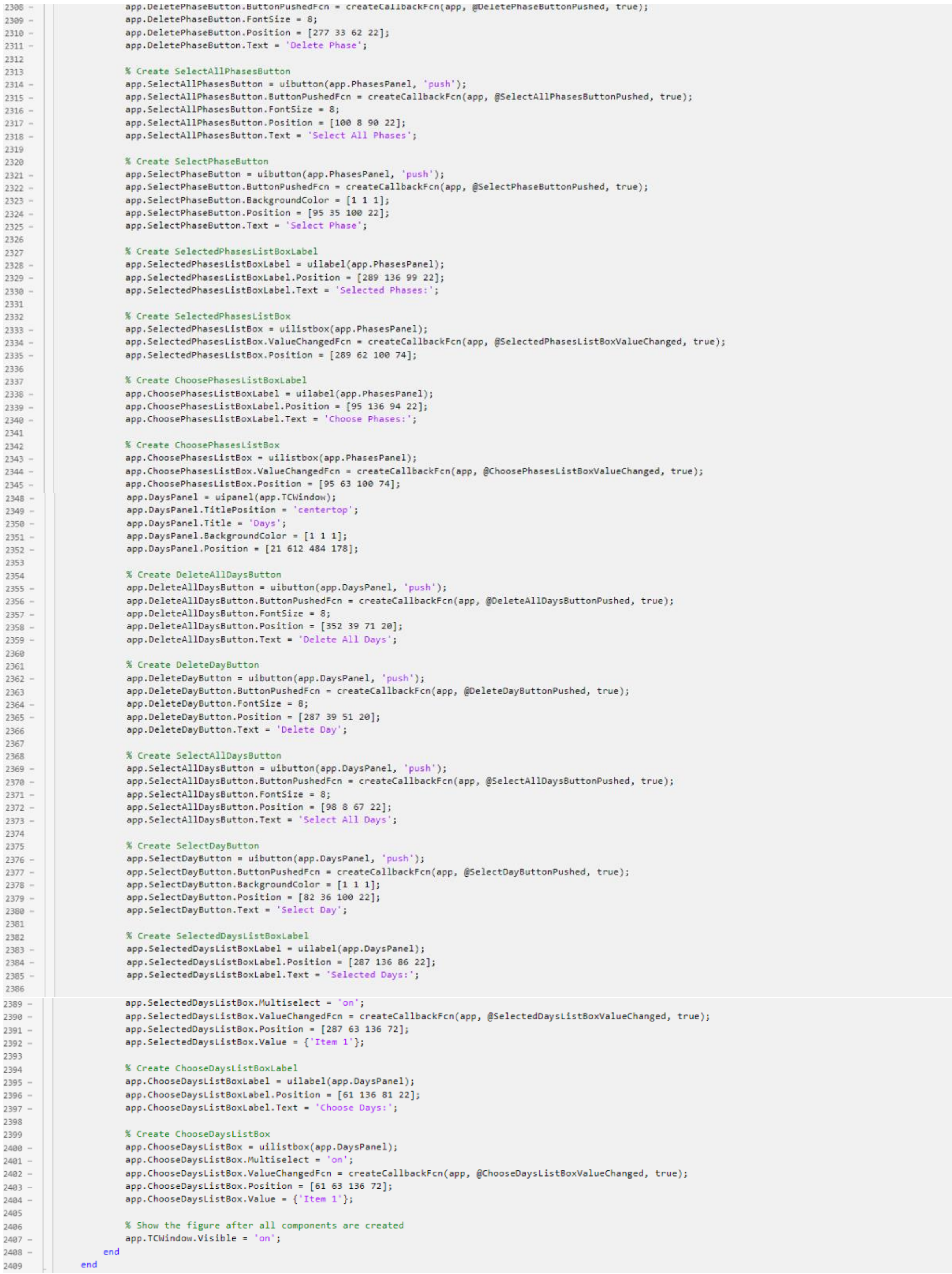


```
            if ~isempty(app.all_sentences_control)
                app.callerApp_sent_control = TFG_Sentences_Control(app.all_sentences_control);
            end
            if ~isempty(app.all_sentences_perturbation)
                app.callerApp_sent_perturbation = TFG_Sentences_Perturbation(app.all_sentences_perturbation);
            end
            if ~isempty(app.all_sentences_washout)
                app.callerApp_sent_washout = TFG_Sentences_Washout(app.all_sentences_washout);
            end

        end

        % Value changed function: SaveScreenshotButton
        function SaveScreenshotButtonValueChanged(app, event)

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

        end
    end
```

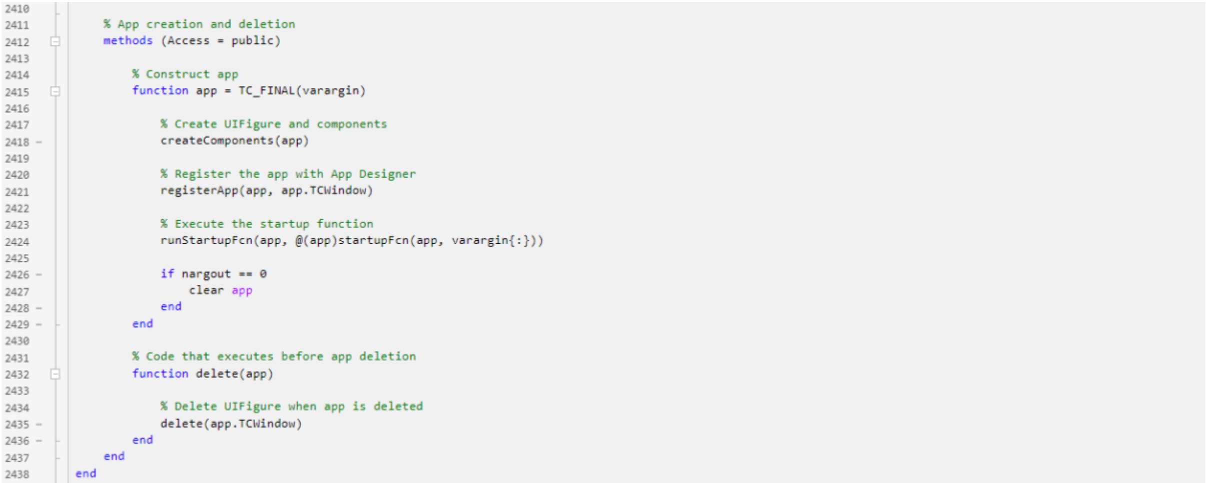
Component initialization



```
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create TCWindow and hide until all components are created
            app.TCWindow = uifigure('Visible', 'off');
            app.TCWindow.Position = [0 40 1536 800];
            app.TCWindow.Name = 'MATLAB App';

            % Create TargetCellPanel
            app.TargetCellPanel = uipanel(app.TCWindow);
            app.TargetCellPanel.TitlePosition = 'centertop';
            app.TargetCellPanel.Title = 'Target Cell';
            app.TargetCellPanel.BackgroundColor = [1 1 1];
            app.TargetCellPanel.Position = [1031 612 484 178];

            % Create SaveScreenshotButton
            app.SaveScreenshotButton = uibutton(app.TargetCellPanel, 'state');
            app.SaveScreenshotButton.ValueChangedFcn = createCallbackFcn(app, @SaveScreenshotButtonValueChanged, true);
            app.SaveScreenshotButton.Text = 'Save Screenshot';
            app.SaveScreenshotButton.FontWeight = 'bold';
            app.SaveScreenshotButton.Position = [310 18 112 22];

            % Create SelectTCButton
            app.SelectTCButton = uibutton(app.TargetCellPanel, 'push');
            app.SelectTCButton.ButtonPushedFcn = createCallbackFcn(app, @SelectTCButtonPushed, true);
            app.SelectTCButton.Position = [58 8 100 22];
            app.SelectTCButton.Text = 'Select TC';

            % Create ShowTargetCellOptionsButton
            app.ShowTargetCellOptionsButton = uibutton(app.TargetCellPanel, 'push');
            app.ShowTargetCellOptionsButton.ButtonPushedFcn = createCallbackFcn(app, @ShowTargetCellOptionsButtonPushed, true);
            app.ShowTargetCellOptionsButton.Position = [36 130 151 22];
            app.ShowTargetCellOptionsButton.Text = 'Show Target Cell Options';

            % Create SelectedTCTextAreaLabel
            app.SelectedTCTextAreaLabel = uilabel(app.TargetCellPanel);
            app.SelectedTCTextAreaLabel.HorizontalAlignment = 'right';
            app.SelectedTCTextAreaLabel.Position = [274 83 75 22];
            app.SelectedTCTextAreaLabel.Text = 'Selected TC:';
            app.SelectedTCTextArea = uitextarea(app.TargetCellPanel);
            app.SelectedTCTextArea.Position = [353 80 105 25];

            % Create NumTCAvailableTextAreaLabel
            app.NumTCAvailableTextAreaLabel = uilabel(app.TargetCellPanel);
            app.NumTCAvailableTextAreaLabel.HorizontalAlignment = 'right';
            app.NumTCAvailableTextAreaLabel.Position = [274 114 105 22];
            app.NumTCAvailableTextAreaLabel.Text = 'Num. TC Available';

            % Create NumTCAvailableTextArea
            app.NumTCAvailableTextArea = uitextarea(app.TargetCellPanel);
            app.NumTCAvailableTextArea.Editable = 'off';
            app.NumTCAvailableTextArea.Position = [387 113 33 24];

            % Create ChooseTargetCellListBoxLabel
            app.ChooseTargetCellListBoxLabel = uilabel(app.TargetCellPanel);
            app.ChooseTargetCellListBoxLabel.Position = [47 105 111 22];
            app.ChooseTargetCellListBoxLabel.Text = 'Choose Target Cell:';

            % Create ChooseTargetCellListBox
            app.ChooseTargetCellListBox = uilistbox(app.TargetCellPanel);
            app.ChooseTargetCellListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseTargetCellListBoxValueChanged, true);
            app.ChooseTargetCellListBox.Position = [47 39 129 66];

            % Create PhasesPanel
            app.PhasesPanel = uipanel(app.TCWindow);
            app.PhasesPanel.TitlePosition = 'centertop';
            app.PhasesPanel.Title = 'Phases';
            app.PhasesPanel.BackgroundColor = [1 1 1];
            app.PhasesPanel.Position = [526 612 484 178];

            % Create DeleteAllPhasesButton
            app.DeleteAllPhasesButton = uibutton(app.PhasesPanel, 'push');
            app.DeleteAllPhasesButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllPhasesButtonPushed, true);
            app.DeleteAllPhasesButton.FontSize = 8;
            app.DeleteAllPhasesButton.Position = [348 33 76 22];
            app.DeleteAllPhasesButton.Text = 'Delete All Phases';

```



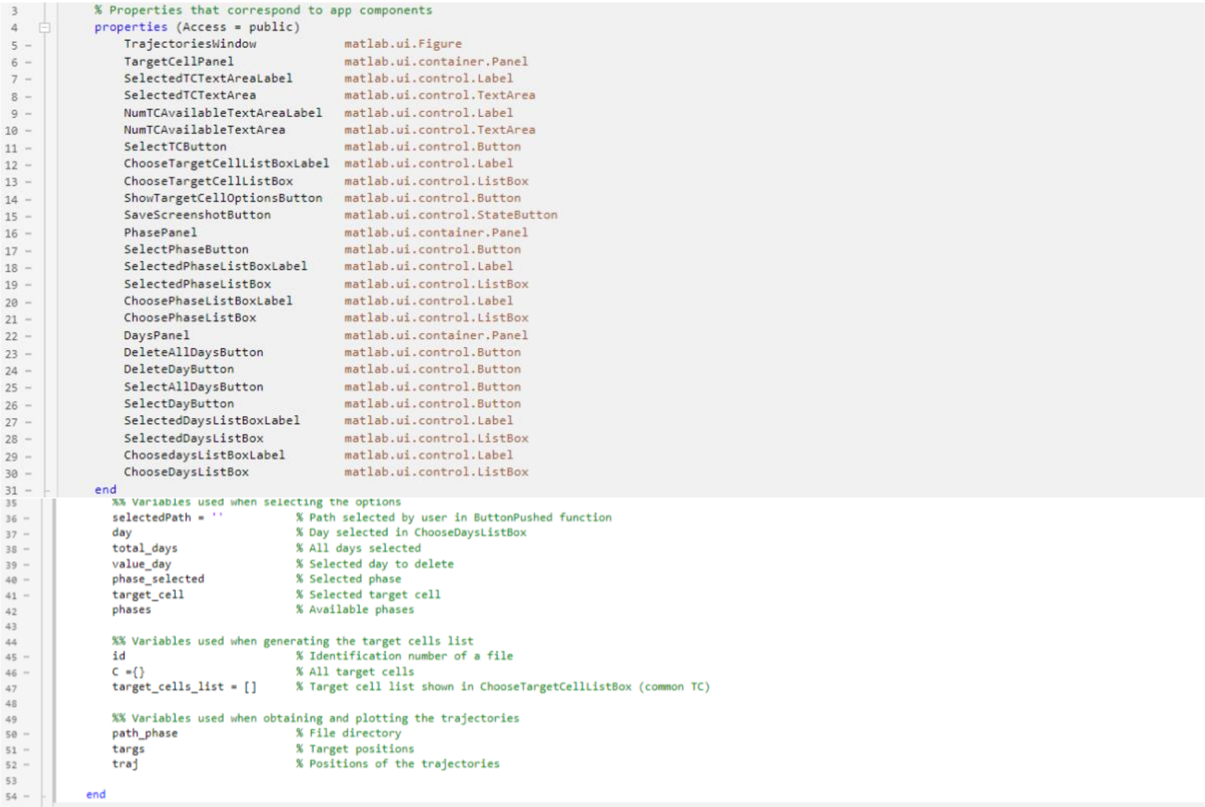
```
            app.DeletePhaseButton.ButtonPushedFcn = createCallbackFcn(app, @DeletePhaseButtonPushed, true);
            app.DeletePhaseButton.FontSize = 8;
            app.DeletePhaseButton.Position = [277 33 62 22];
            app.DeletePhaseButton.Text = 'Delete Phase';

            % Create SelectAllPhasesButton
            app.SelectAllPhasesButton = uibutton(app.PhasesPanel, 'push');
            app.SelectAllPhasesButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllPhasesButtonPushed, true);
            app.SelectAllPhasesButton.FontSize = 8;
            app.SelectAllPhasesButton.Position = [100 8 90 22];
            app.SelectAllPhasesButton.Text = 'Select All Phases';

            % Create SelectPhaseButton
            app.SelectPhaseButton = uibutton(app.PhasesPanel, 'push');
            app.SelectPhaseButton.ButtonPushedFcn = createCallbackFcn(app, @SelectPhaseButtonPushed, true);
            app.SelectPhaseButton.BackgroundColor = [1 1 1];
            app.SelectPhaseButton.Position = [95 35 100 22];
            app.SelectPhaseButton.Text = 'Select Phase';

            % Create SelectedPhasesListBoxLabel
            app.SelectedPhasesListBoxLabel = uilabel(app.PhasesPanel);
            app.SelectedPhasesListBoxLabel.Position = [289 136 99 22];
            app.SelectedPhasesListBoxLabel.Text = 'Selected Phases:';

            % Create SelectedPhasesListBox
            app.SelectedPhasesListBox = uilistbox(app.PhasesPanel);
            app.SelectedPhasesListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedPhasesListBoxValueChanged, true);
            app.SelectedPhasesListBox.Position = [289 62 100 74];

            % Create ChoosePhasesListBoxLabel
            app.ChoosePhasesListBoxLabel = uilabel(app.PhasesPanel);
            app.ChoosePhasesListBoxLabel.Position = [95 136 94 22];
            app.ChoosePhasesListBoxLabel.Text = 'Choose Phases:';

            % Create ChoosePhasesListBox
            app.ChoosePhasesListBox = uilistbox(app.PhasesPanel);
            app.ChoosePhasesListBox.ValueChangedFcn = createCallbackFcn(app, @ChoosePhasesListBoxValueChanged, true);
            app.ChoosePhasesListBox.Position = [95 63 100 74];
            app.DaysPanel = uipanel(app.TCWindow);
            app.DaysPanel.TitlePosition = 'centertop';
            app.DaysPanel.Title = 'Days';
            app.DaysPanel.BackgroundColor = [1 1 1];
            app.DaysPanel.Position = [21 612 484 178];

            % Create DeleteAllDaysButton
            app.DeleteAllDaysButton = uibutton(app.DaysPanel, 'push');
            app.DeleteAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllDaysButtonPushed, true);
            app.DeleteAllDaysButton.FontSize = 8;
            app.DeleteAllDaysButton.Position = [352 39 71 20];
            app.DeleteAllDaysButton.Text = 'Delete All Days';

            % Create DeleteDayButton
            app.DeleteDayButton = uibutton(app.DaysPanel, 'push');
            app.DeleteDayButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteDayButtonPushed, true);
            app.DeleteDayButton.FontSize = 8;
            app.DeleteDayButton.Position = [287 39 51 20];
            app.DeleteDayButton.Text = 'Delete Day';

            % Create SelectAllDaysButton
            app.SelectAllDaysButton = uibutton(app.DaysPanel, 'push');
            app.SelectAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllDaysButtonPushed, true);
            app.SelectAllDaysButton.FontSize = 8;
            app.SelectAllDaysButton.Position = [98 8 67 22];
            app.SelectAllDaysButton.Text = 'Select All Days';

            % Create SelectDayButton
            app.SelectDayButton = uibutton(app.DaysPanel, 'push');
            app.SelectDayButton.ButtonPushedFcn = createCallbackFcn(app, @SelectDayButtonPushed, true);
            app.SelectDayButton.BackgroundColor = [1 1 1];
            app.SelectDayButton.Position = [82 36 100 22];
            app.SelectDayButton.Text = 'Select Day';

            % Create SelectedDaysListBoxLabel
            app.SelectedDaysListBoxLabel = uilabel(app.DaysPanel);
            app.SelectedDaysListBoxLabel.Position = [287 136 86 22];
            app.SelectedDaysListBoxLabel.Text = 'Selected Days:';

            app.SelectedDaysListBox.Multiselect = 'on';
            app.SelectedDaysListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedDaysListBoxValueChanged, true);
            app.SelectedDaysListBox.Position = [287 63 136 72];
            app.SelectedDaysListBox.Value = {'Item 1'};

            % Create ChooseDaysListBoxLabel
            app.ChooseDaysListBoxLabel = uilabel(app.DaysPanel);
            app.ChooseDaysListBoxLabel.Position = [61 136 81 22];
            app.ChooseDaysListBoxLabel.Text = 'Choose Days:';

            % Create ChooseDaysListBox
            app.ChooseDaysListBox = uilistbox(app.DaysPanel);
            app.ChooseDaysListBox.Multiselect = 'on';
            app.ChooseDaysListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseDaysListBoxValueChanged, true);
            app.ChooseDaysListBox.Position = [61 63 136 72];
            app.ChooseDaysListBox.Value = {'Item 1'};

            % Show the figure after all components are created
            app.TCWindow.Visible = 'on';
        end
    end
```



```

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = TC_FINAL(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TCWindow)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TCWindow)
        end
    end
end
```

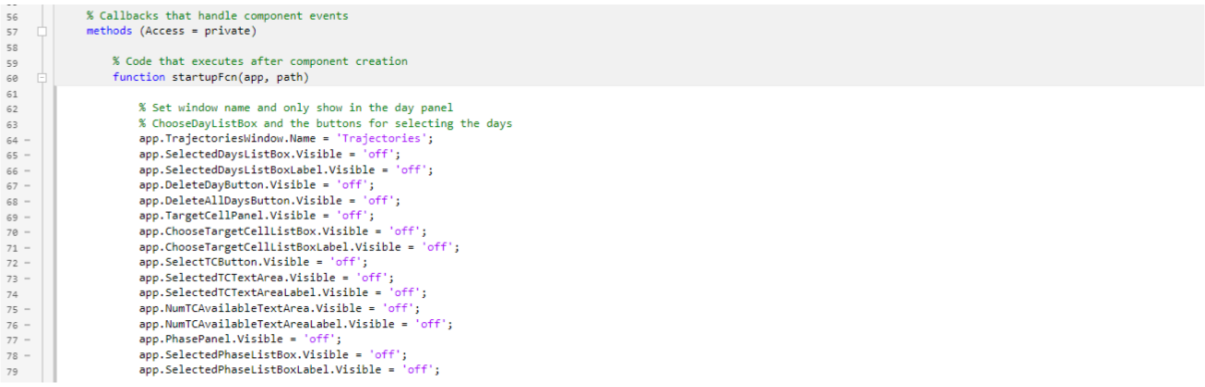
ANNEX III. Code used for the development of the Trajectories window

Properties

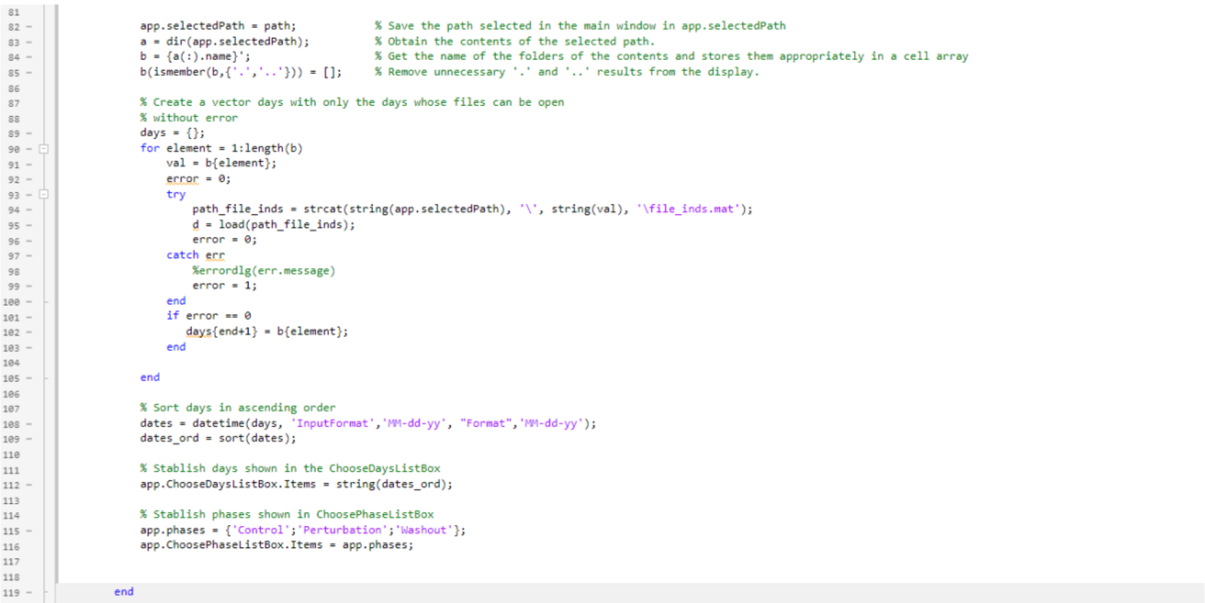


```
    % Properties that correspond to app components
    properties (Access = public)
        TrajectoriesWindow            matlab.ui.Figure
        TargetCellPanel               matlab.ui.container.Panel
        SelectedTCTextAreaLabel       matlab.ui.control.Label
        SelectedTCTextArea            matlab.ui.control.TextArea
        NumTCAvailableTextAreaLabel   matlab.ui.control.Label
        NumTCAvailableTextArea        matlab.ui.control.TextArea
        SelectTCButton                matlab.ui.control.Button
        ChooseTargetCellListBoxLabel  matlab.ui.control.Label
        ChooseTargetCellListBox       matlab.ui.control.ListBox
        ShowTargetCellOptionsButton   matlab.ui.control.Button
        SaveScreenshotButton          matlab.ui.control.StateButton
        PhasePanel                    matlab.ui.container.Panel
        SelectPhaseButton             matlab.ui.control.Button
        SelectedPhaseListBoxLabel     matlab.ui.control.Label
        SelectedPhaseListBox          matlab.ui.control.ListBox
        ChoosePhaseListBoxLabel       matlab.ui.control.Label
        ChoosePhaseListBox            matlab.ui.control.ListBox
        DaysPanel                     matlab.ui.container.Panel
        DeleteAllDaysButton           matlab.ui.control.Button
        DeleteDayButton               matlab.ui.control.Button
        SelectAllDaysButton           matlab.ui.control.Button
        SelectDayButton               matlab.ui.control.Button
        SelectedDaysListBoxLabel      matlab.ui.control.Label
        SelectedDaysListBox           matlab.ui.control.ListBox
        ChoosedaysListBoxLabel        matlab.ui.control.Label
        ChooseDaysListBox             matlab.ui.control.ListBox
    end
        %% Variables used when selecting the options
        selectedPath = ''           % Path selected by user in ButtonPushed function
        day                         % Day selected in ChooseDaysListBox
        total_days                  % All days selected
        value_day                   % Selected day to delete
        phase_selected              % Selected phase
        target_cell                 % Selected target cell
        phases                      % Available phases

        %% Variables used when generating the target cells list
        id                          % Identification number of a file
        C ={}                       % All target cells
        target_cells_list = []      % Target cell list shown in ChooseTargetCellListBox (common TC)

        %% Variables used when obtaining and plotting the trajectories
        path_phase                  % File directory
        targs                       % Target positions
        traj                        % Positions of the trajectories

    end
```

Startup function

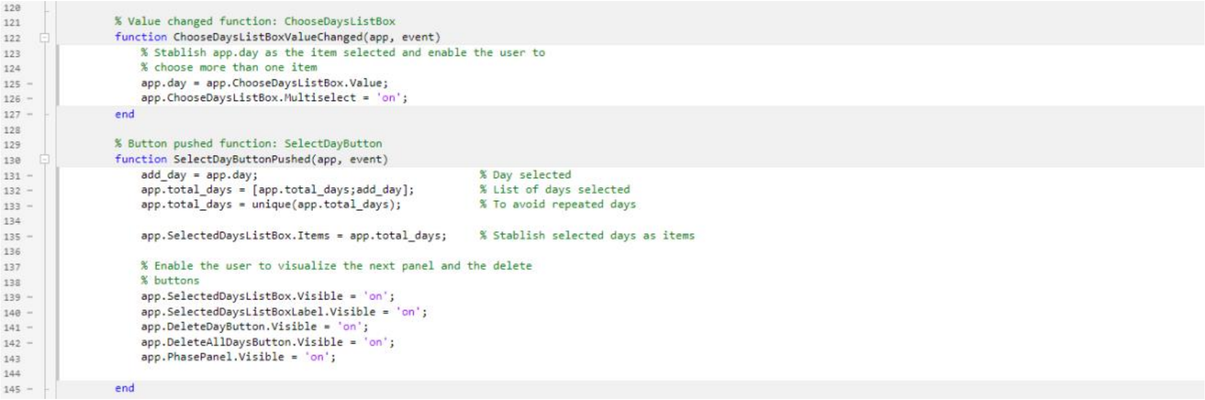


```
    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app, path)

            % Set window name and only show in the day panel
            % ChooseDayListBox and the buttons for selecting the days
            app.TrajectoriesWindow.Name = 'Trajectories';
            app.SelectedDaysListBox.Visible = 'off';
            app.SelectedDaysListBoxLabel.Visible = 'off';
            app.DeleteDayButton.Visible = 'off';
            app.DeleteAllDaysButton.Visible = 'off';
            app.TargetCellPanel.Visible = 'off';
            app.ChooseTargetCellListBox.Visible = 'off';
            app.ChooseTargetCellListBoxLabel.Visible = 'off';
            app.SelectTCButton.Visible = 'off';
            app.SelectedTCTextArea.Visible = 'off';
            app.SelectedTCTextAreaLabel.Visible = 'off';
            app.NumTCAvailableTextArea.Visible = 'off';
            app.NumTCAvailableTextAreaLabel.Visible = 'off';
            app.PhasePanel.Visible = 'off';
            app.SelectedPhaseListBox.Visible = 'off';
            app.SelectedPhaseListBoxLabel.Visible = 'off';
```

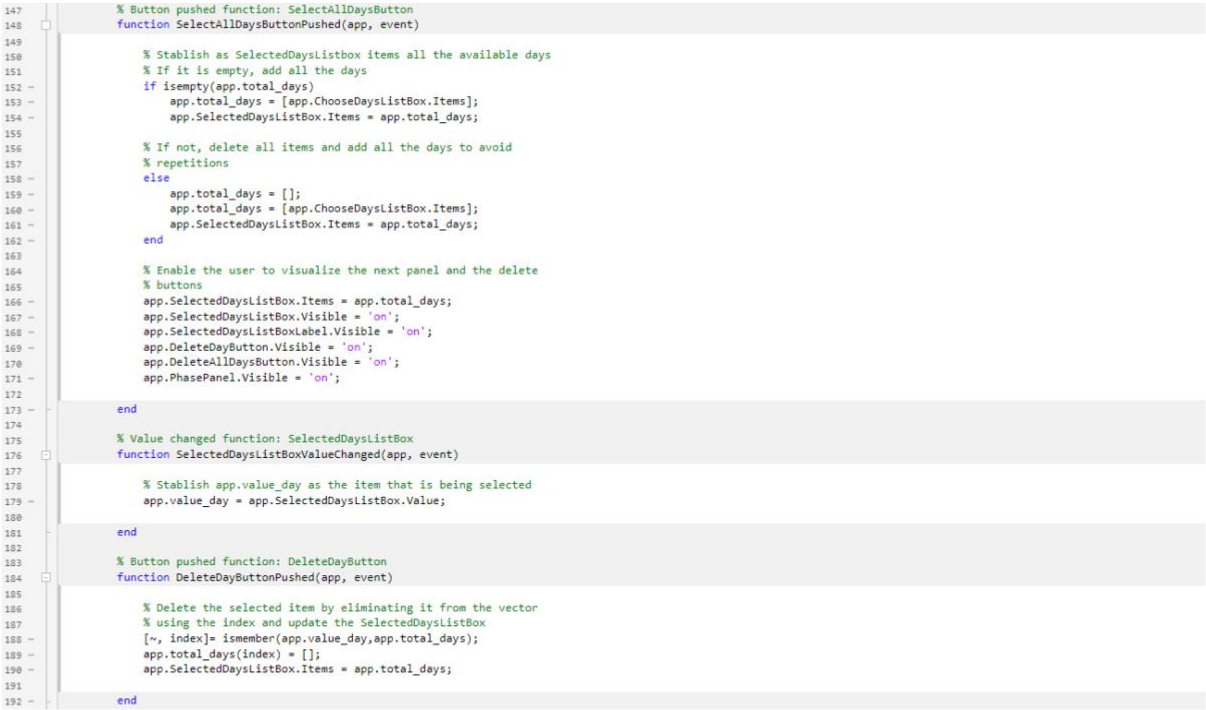


```

            app.selectedPath = path;            % Save the path selected in the main window in app.selectedPath
            a = dir(app.selectedPath);          % Obtain the contents of the selected path.
            b = {a(:).name}';                   % Get the name of the folders of the contents and stores them appropriately in a cell array
            b(ismember(b,{'.','..'})) = [];     % Remove unnecessary '.' and '..' results from the display.

            % Create a vector days with only the days whose files can be open
            % without error
            days = {};
            for element = 1:length(b)
                val = b{element};
                error = 0;
                try
                    path_file_inds = strcat(string(app.selectedPath), '\', string(val), '\file_inds.mat');
                    d = load(path_file_inds);
                    error = 0;
                catch err
                    %errordlg(err.message)
                    error = 1;
                end
                if error == 0
                   days{end+1} = b{element};
                end

            end

            % Sort days in ascending order
            dates = datetime(days, 'InputFormat','MM-dd-yy', "Format",'MM-dd-yy');
            dates_ord = sort(dates);

            % Stablish days shown in the ChooseDaysListBox
            app.ChooseDaysListBox.Items = string(dates_ord);

            % Stablish phases shown in ChoosePhaseListBox
            app.phases = {'Control';'Perturbation';'Washout'};
            app.ChoosePhaseListBox.Items = app.phases;


        end
```

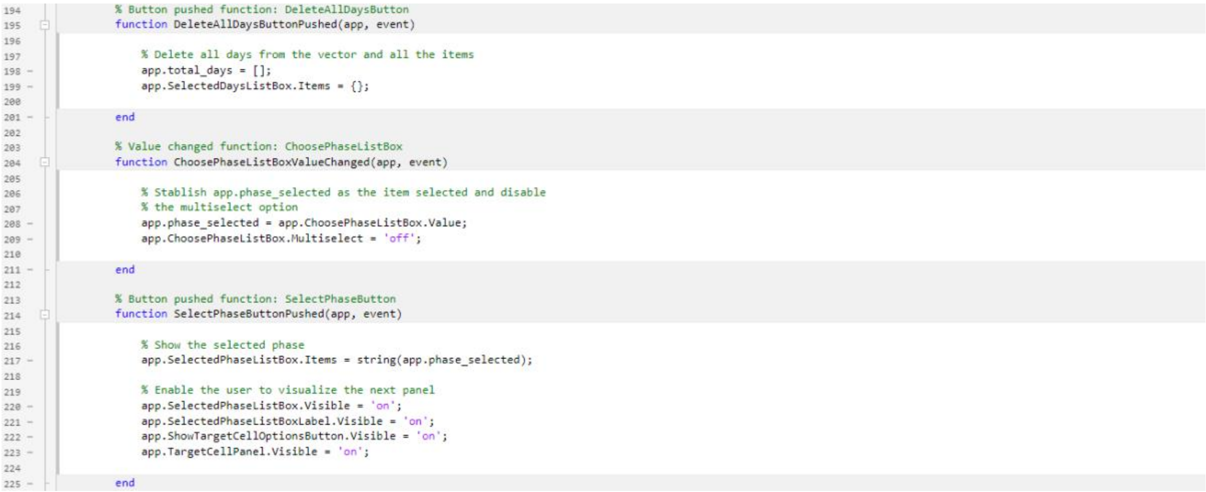
Callback functions



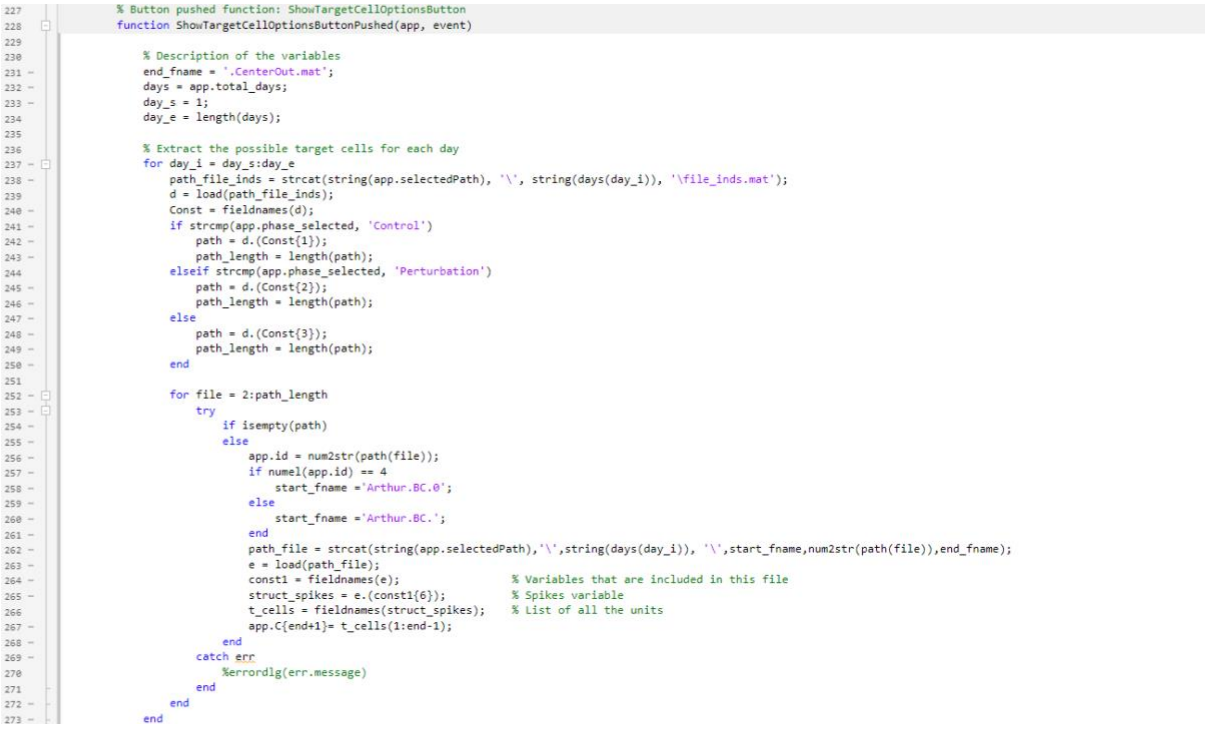
```

        % Value changed function: ChooseDaysListBox
        function ChooseDaysListBoxValueChanged(app, event)
            % Stablish app.day as the item selected and enable the user to
            % choose more than one item
            app.day = app.ChooseDaysListBox.Value;
            app.ChooseDaysListBox.Multiselect = 'on';
        end

        % Button pushed function: SelectDayButton
        function SelectDayButtonPushed(app, event)
            add_day = app.day;                                  % Day selected
            app.total_days = [app.total_days;add_day];          % List of days selected
            app.total_days = unique(app.total_days);            % To avoid repeated days

            app.SelectedDaysListBox.Items = app.total_days;     % Stablish selected days as items

            % Enable the user to visualize the next panel and the delete
            % buttons
            app.SelectedDaysListBox.Visible = 'on';
            app.SelectedDaysListBoxLabel.Visible = 'on';
            app.DeleteDayButton.Visible = 'on';
            app.DeleteAllDaysButton.Visible = 'on';
            app.PhasePanel.Visible = 'on';

        end
```



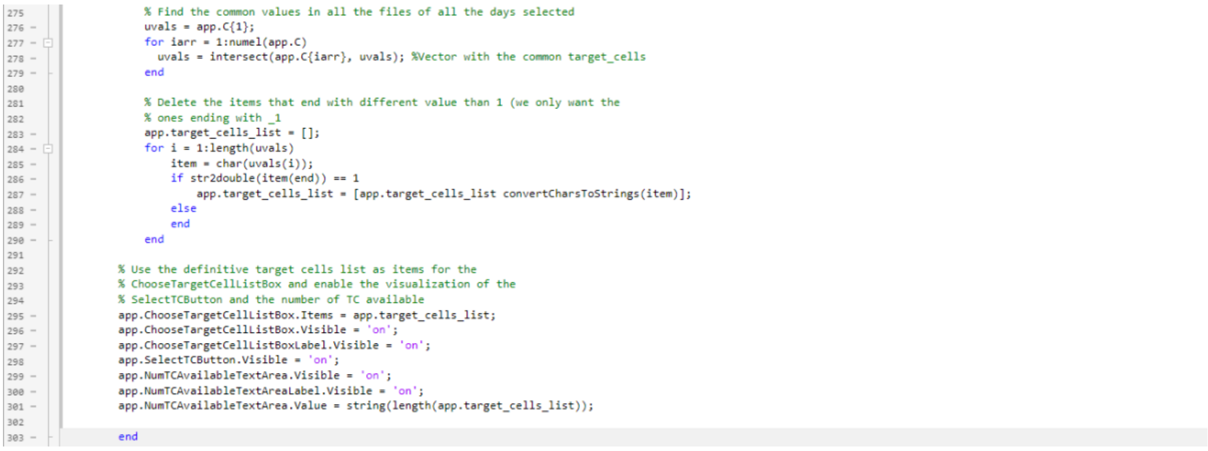
```
        % Button pushed function: SelectAllDaysButton
        function SelectAllDaysButtonPushed(app, event)

            % Stablish as SelectedDaysListbox items all the available days
            % If it is empty, add all the days
            if isempty(app.total_days)
                app.total_days = [app.ChooseDaysListBox.Items];
                app.SelectedDaysListBox.Items = app.total_days;

            % If not, delete all items and add all the days to avoid
            % repetitions
            else
                app.total_days = [];
                app.total_days = [app.ChooseDaysListBox.Items];
                app.SelectedDaysListBox.Items = app.total_days;
            end

            % Enable the user to visualize the next panel and the delete
            % buttons
            app.SelectedDaysListBox.Items = app.total_days;
            app.SelectedDaysListBox.Visible = 'on';
            app.SelectedDaysListBoxLabel.Visible = 'on';
            app.DeleteDayButton.Visible = 'on';
            app.DeleteAllDaysButton.Visible = 'on';
            app.PhasePanel.Visible = 'on';

        end

        % Value changed function: SelectedDaysListBox
        function SelectedDaysListBoxValueChanged(app, event)

            % Stablish app.value_day as the item that is being selected
            app.value_day = app.SelectedDaysListBox.Value;

        end

        % Button pushed function: DeleteDayButton
        function DeleteDayButtonPushed(app, event)

            % Delete the selected item by eliminating it from the vector
            % using the index and update the SelectedDaysListBox
            [~, index]= ismember(app.value_day,app.total_days);
            app.total_days(index) = [];
            app.SelectedDaysListBox.Items = app.total_days;

        end
```



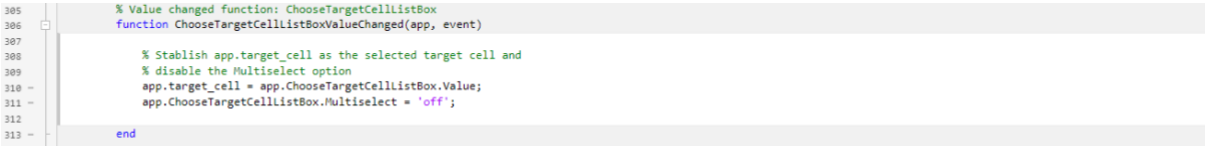
```
        % Button pushed function: DeleteAllDaysButton
        function DeleteAllDaysButtonPushed(app, event)

            % Delete all days from the vector and all the items
            app.total_days = [];
            app.SelectedDaysListBox.Items = {};

        end

        % Value changed function: ChoosePhaseListBox
        function ChoosePhaseListBoxValueChanged(app, event)

            % Stablish app.phase_selected as the item selected and disable
            % the multiselect option
            app.phase_selected = app.ChoosePhaseListBox.Value;
            app.ChoosePhaseListBox.Multiselect = 'off';

        end

        % Button pushed function: SelectPhaseButton
        function SelectPhaseButtonPushed(app, event)

            % Show the selected phase
            app.SelectedPhaseListBox.Items = string(app.phase_selected);

            % Enable the user to visualize the next panel
            app.SelectedPhaseListBox.Visible = 'on';
            app.SelectedPhaseListBoxLabel.Visible = 'on';
            app.ShowTargetCellOptionsButton.Visible = 'on';
            app.TargetCellPanel.Visible = 'on';

        end
```



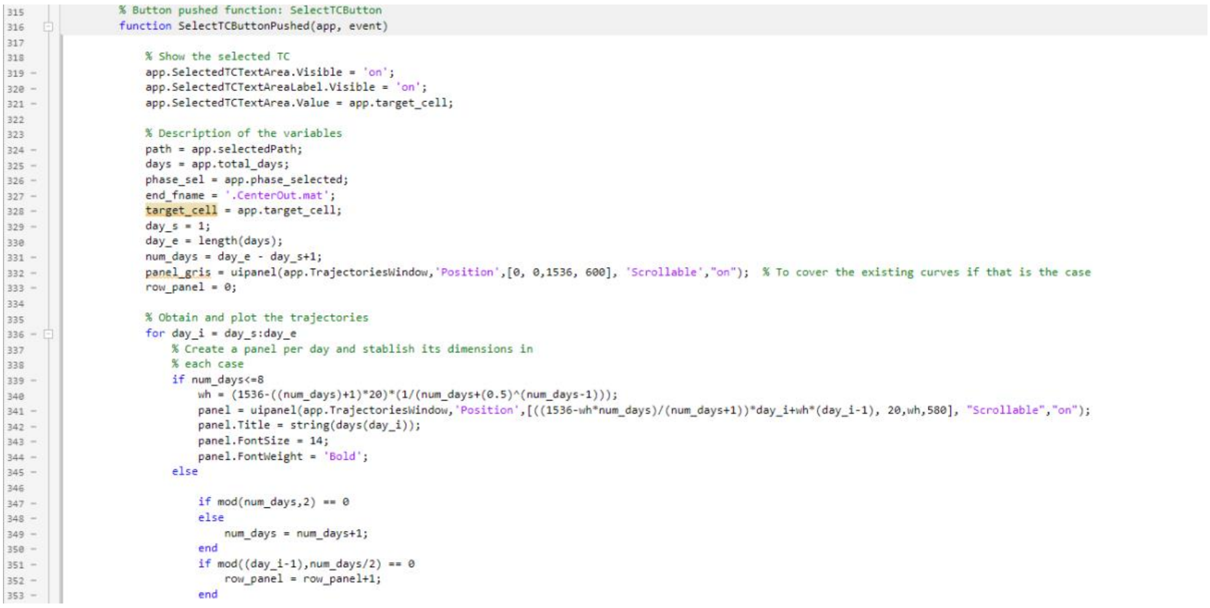
```
% Button pushed function: ShowTargetCellOptionsButton
function ShowTargetCellOptionsButtonPushed(app, event)

    % Description of the variables
    end_fname = '.CenterOut.mat';
    days = app.total_days;
    day_s = 1;
    day_e = length(days);

    % Extract the possible target cells for each day
    for day_i = day_s:day_e
        path_file_inds = strcat(string(app.selectedPath), '\', string(days(day_i)), '\file_inds.mat');
        d = load(path_file_inds);
        Const = fieldnames(d);
        if strcmp(app.phase_selected, 'Control')
            path = d.(Const{1});
            path_length = length(path);
        elseif strcmp(app.phase_selected, 'Perturbation')
            path = d.(Const{2});
            path_length = length(path);
        else
            path = d.(Const{3});
            path_length = length(path);
        end

        for file = 2:path_length
            try
                if isempty(path)
                else
                    app.id = num2str(path(file));
                    if numel(app.id) == 4
                        start_fname ='Arthur.BC.0';
                    else
                        start_fname ='Arthur.BC.';
                    end
                    path_file = strcat(string(app.selectedPath),'\',string(days(day_i)), '\',start_fname,num2str(path(file)),end_fname);
                    e = load(path_file);
                    const1 = fieldnames(e);                 % Variables that are included in this file
                    struct_spikes = e.(const1{6});          % Spikes variable
                    t_cells = fieldnames(struct_spikes);    % List of all the units
                    app.C{end+1}= t_cells(1:end-1);
                end
            catch err
                %errordlg(err.message)
            end
        end
    end
```



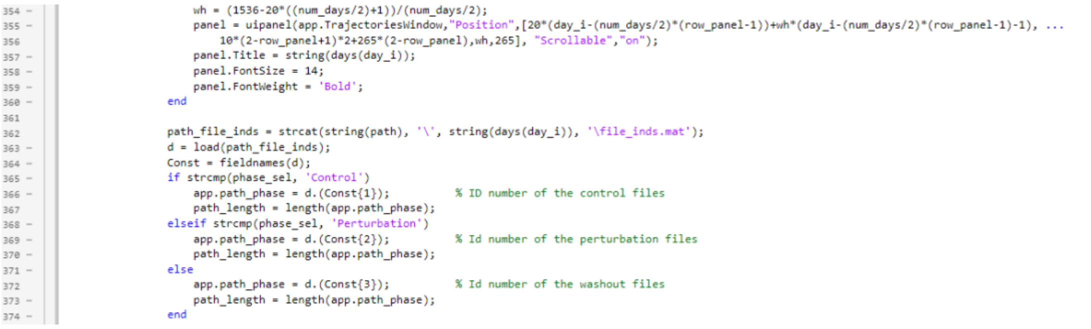
```
    % Find the common values in all the files of all the days selected
    uvals = app.C{1};
    for iarr = 1:numel(app.C)
      uvals = intersect(app.C{iarr}, uvals); %Vector with the common target_cells
    end

    % Delete the items that end with different value than 1 (we only want the
    % ones ending with _1
    app.target_cells_list = [];
    for i = 1:length(uvals)
        item = char(uvals(i));
        if str2double(item(end)) == 1
            app.target_cells_list = [app.target_cells_list convertCharsToStrings(item)];
        else
        end
    end

% Use the definitive target cells list as items for the
% ChooseTargetCellListBox and enable the visualization of the
% SelectTCButton and the number of TC available
app.ChooseTargetCellListBox.Items = app.target_cells_list;
app.ChooseTargetCellListBox.Visible = 'on';
app.ChooseTargetCellListBoxLabel.Visible = 'on';
app.SelectTCButton.Visible = 'on';
app.NumTCAvailableTextArea.Visible = 'on';
app.NumTCAvailableTextAreaLabel.Visible = 'on';
app.NumTCAvailableTextArea.Value = string(length(app.target_cells_list));

end
```



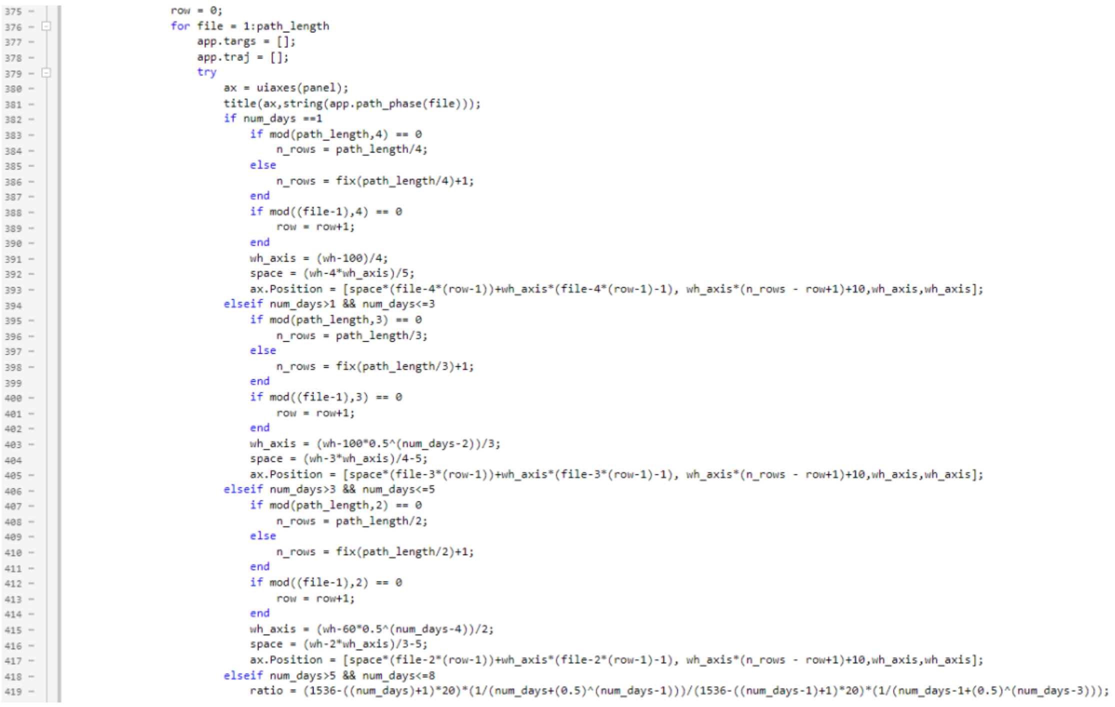
```
% Value changed function: ChooseTargetCellListBox
function ChooseTargetCellListBoxValueChanged(app, event)

    % Stablish app.target_cell as the selected target cell and
    % disable the Multiselect option
    app.target_cell = app.ChooseTargetCellListBox.Value;
    app.ChooseTargetCellListBox.Multiselect = 'off';

end
```



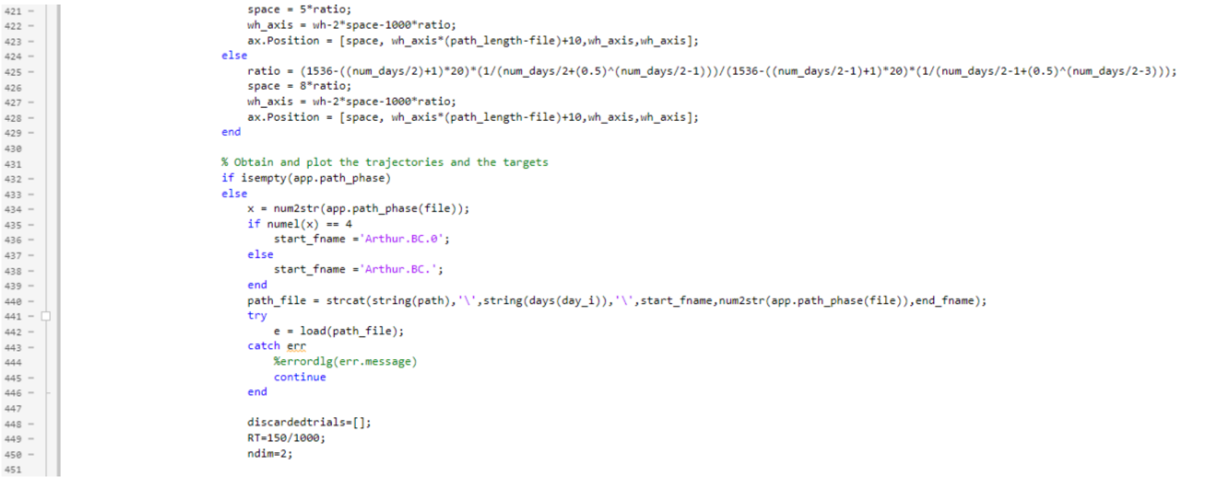
```
        % Button pushed function: SelectTCButton
        function SelectTCButtonPushed(app, event)

            % Show the selected TC
            app.SelectedTCTextArea.Visible = 'on';
            app.SelectedTCTextAreaLabel.Visible = 'on';
            app.SelectedTCTextArea.Value = app.target_cell;

            % Description of the variables
            path = app.selectedPath;
            days = app.total_days;
            phase_sel = app.phase_selected;
            end_fname = '.CenterOut.mat';
            target_cell = app.target_cell;
            day_s = 1;
            day_e = length(days);
            num_days = day_e - day_s+1;
            panel_gris = uipanel(app.TrajectoriesWindow,'Position',[0, 0,1536, 600], 'Scrollable',"on");  % To cover the existing curves if that is the case
            row_panel = 0;

            % Obtain and plot the trajectories
            for day_i = day_s:day_e
                % Create a panel per day and stablish its dimensions in
                % each case
                if num_days<=8
                    wh = (1536-((num_days)+1)*20)*(1/(num_days+(0.5)^(num_days-1)));
                    panel = uipanel(app.TrajectoriesWindow,'Position',[((1536-wh*num_days)/(num_days+1))*day_i+wh*(day_i-1), 20,wh,580], "Scrollable","on");
                    panel.Title = string(days(day_i));
                    panel.FontSize = 14;
                    panel.FontWeight = 'Bold';
                else

                    if mod(num_days,2) == 0
                    else
                        num_days = num_days+1;
                    end
                    if mod((day_i-1),num_days/2) == 0
                        row_panel = row_panel+1;
                    end
```



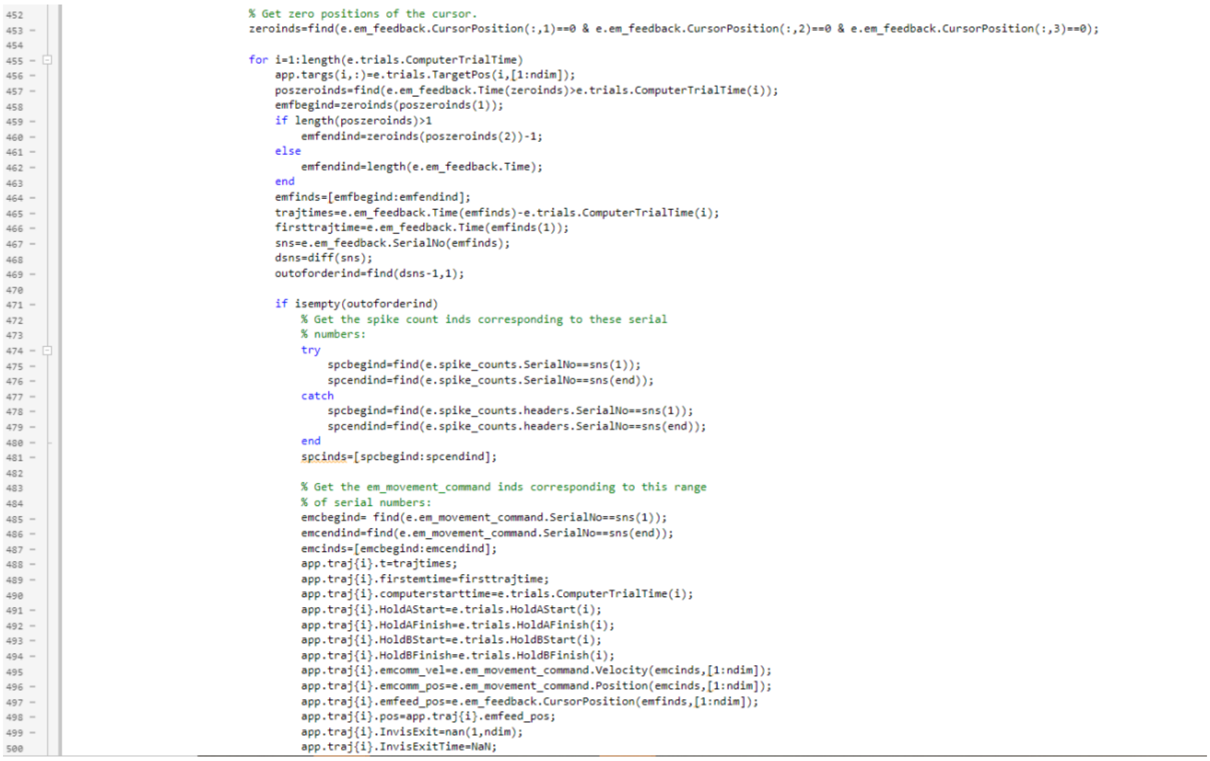
```
                    wh = (1536-20*((num_days/2)+1))/(num_days/2);
                    panel = uipanel(app.TrajectoriesWindow,"Position",[20*(day_i-(num_days/2)*(row_panel-1))+wh*(day_i-(num_days/2)*(row_panel-1)-1), ...
                        10*(2-row_panel+1)*2+265*(2-row_panel),wh,265], "Scrollable","on");
                    panel.Title = string(days(day_i));
                    panel.FontSize = 14;
                    panel.FontWeight = 'Bold';
                end

                path_file_inds = strcat(string(path), '\', string(days(day_i)), '\file_inds.mat');
                d = load(path_file_inds);
                Const = fieldnames(d);
                if strcmp(phase_sel, 'Control')
                    app.path_phase = d.(Const{1});          % ID number of the control files
                    path_length = length(app.path_phase);
                elseif strcmp(phase_sel, 'Perturbation')
                    app.path_phase = d.(Const{2});          % Id number of the perturbation files
                    path_length = length(app.path_phase);
                else
                    app.path_phase = d.(Const{3});          % Id number of the washout files
                    path_length = length(app.path_phase);
                end
```



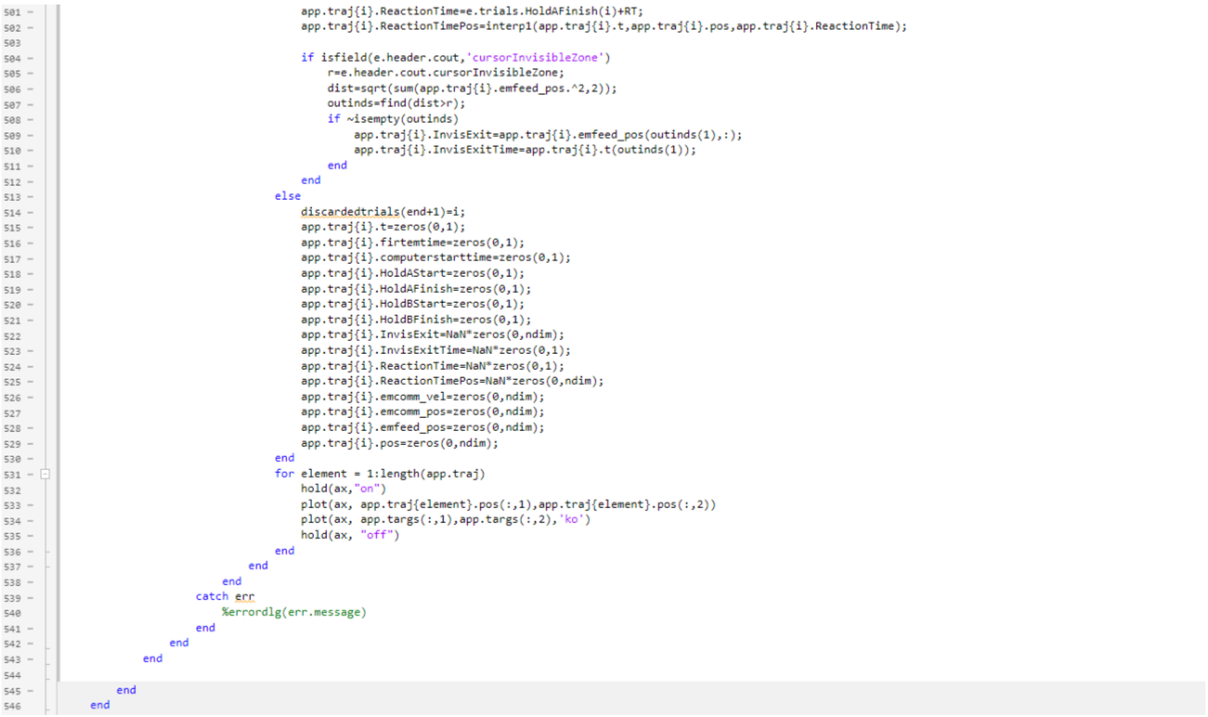
```
                row = 0;
                for file = 1:path_length
                    app.targs = [];
                    app.traj = [];
                    try
                        ax = uiaxes(panel);
                        title(ax,string(app.path_phase(file)));
                        if num_days ==1
                            if mod(path_length,4) == 0
                                n_rows = path_length/4;
                            else
                                n_rows = fix(path_length/4)+1;
                            end
                            if mod((file-1),4) == 0
                                row = row+1;
                            end
                            wh_axis = (wh-100)/4;
                            space = (wh-4*wh_axis)/5;
                            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];
                        elseif num_days>1 && num_days<=3
                            if mod(path_length,3) == 0
                                n_rows = path_length/3;
                            else
                                n_rows = fix(path_length/3)+1;
                            end
                            if mod((file-1),3) == 0
                                row = row+1;
                            end
                            wh_axis = (wh-100*0.5^(num_days-2))/3;
                            space = (wh-3*wh_axis)/4-5;
                            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];
                        elseif num_days>3 && num_days<=5
                            if mod(path_length,2) == 0
                                n_rows = path_length/2;
                            else
                                n_rows = fix(path_length/2)+1;
                            end
                            if mod((file-1),2) == 0
                                row = row+1;
                            end
                            wh_axis = (wh-60*0.5^(num_days-4))/2;
                            space = (wh-2*wh_axis)/3-5;
                            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];
                        elseif num_days>5 && num_days<=8
                            ratio = (1536-((num_days)+1)*20)*(1/(num_days+(0.5)^(num_days-1)))/(1536-((num_days-1)+1)*20)*(1/(num_days-1+(0.5)^(num_days-3)));
```



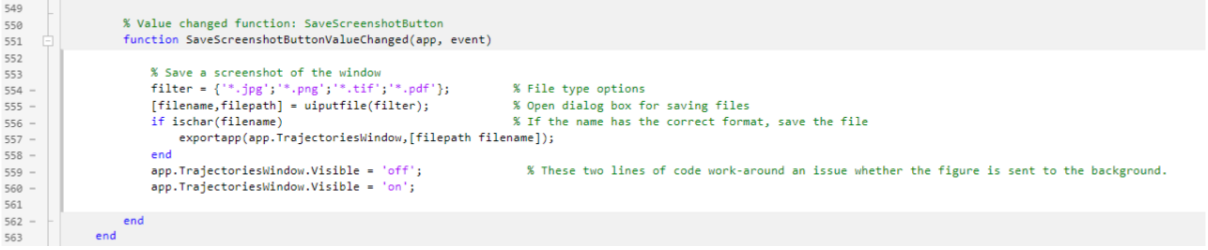
```
    space = 5*ratio;
    wh_axis = wh-2*space-1000*ratio;
    ax.Position = [space, wh_axis*(path_length-file)+10,wh_axis,wh_axis];
else
    ratio = (1536-((num_days/2)+1)*20)*(1/(num_days/2+(0.5)^(num_days/2-1)))/(1536-((num_days/2-1)+1)*20)*(1/(num_days/2-1+(0.5)^(num_days/2-3)));
    space = 8*ratio;
    wh_axis = wh-2*space-1000*ratio;
    ax.Position = [space, wh_axis*(path_length-file)+10,wh_axis,wh_axis];
end

% Obtain and plot the trajectories and the targets
if isempty(app.path_phase)
else
    x = num2str(app.path_phase(file));
    if numel(x) == 4
        start_fname ='Arthur.BC.0';
    else
        start_fname ='Arthur.BC.';
    end
    path_file = strcat(string(path),'\',string(days(day_i)),'\',start_fname,num2str(app.path_phase(file)),end_fname);
    try
        e = load(path_file);
    catch err
        %errordlg(err.message)
        continue
    end

    discardedtrials=[];
    RT=150/1000;
    ndim=2;

```



```
    % Get zero positions of the cursor.
    zeroinds=find(e.em_feedback.CursorPosition(:,1)==0 & e.em_feedback.CursorPosition(:,2)==0 & e.em_feedback.CursorPosition(:,3)==0);

    for i=1:length(e.trials.ComputerTrialTime)
        app.targs(i,:)=e.trials.TargetPos(i,[1:ndim]);
        poszeroinds=find(e.em_feedback.Time(zeroinds)>e.trials.ComputerTrialTime(i));
        emfbegind=zeroinds(poszeroinds(1));
        if length(poszeroinds)>1
            emfendind=zeroinds(poszeroinds(2))-1;
        else
            emfendind=length(e.em_feedback.Time);
        end
        emfinds=[emfbegind:emfendind];
        trajtimes=e.em_feedback.Time(emfinds)-e.trials.ComputerTrialTime(i);
        firsttrajtime=e.em_feedback.Time(emfinds(1));
        sns=e.em_feedback.SerialNo(emfinds);
        dsns=diff(sns);
        outoforderind=find(dsns-1,1);

        if isempty(outoforderind)
            % Get the spike count inds corresponding to these serial
            % numbers:
            try
                spcbegind=find(e.spike_counts.SerialNo==sns(1));
                spcendind=find(e.spike_counts.SerialNo==sns(end));
            catch
                spcbegind=find(e.spike_counts.headers.SerialNo==sns(1));
                spcendind=find(e.spike_counts.headers.SerialNo==sns(end));
            end
            spcinds=[spcbegind:spcendind];

            % Get the em_movement_command inds corresponding to this range
            % of serial numbers:
            emcbegind= find(e.em_movement_command.SerialNo==sns(1));
            emcendind=find(e.em_movement_command.SerialNo==sns(end));
            emcinds=[emcbegind:emcendind];
            app.traj{i}.t=trajtimes;
            app.traj{i}.firstemtime=firsttrajtime;
            app.traj{i}.computerstarttime=e.trials.ComputerTrialTime(i);
            app.traj{i}.HoldAStart=e.trials.HoldAStart(i);
            app.traj{i}.HoldAFinish=e.trials.HoldAFinish(i);
            app.traj{i}.HoldBStart=e.trials.HoldBStart(i);
            app.traj{i}.HoldBFinish=e.trials.HoldBFinish(i);
            app.traj{i}.emcomm_vel=e.em_movement_command.Velocity(emcinds,[1:ndim]);
            app.traj{i}.emcomm_pos=e.em_movement_command.Position(emcinds,[1:ndim]);
            app.traj{i}.emfeed_pos=e.em_feedback.CursorPosition(emfinds,[1:ndim]);
            app.traj{i}.pos=app.traj{i}.emfeed_pos;
            app.traj{i}.InvisExit=nan(1,ndim);
            app.traj{i}.InvisExitTime=NaN;
```

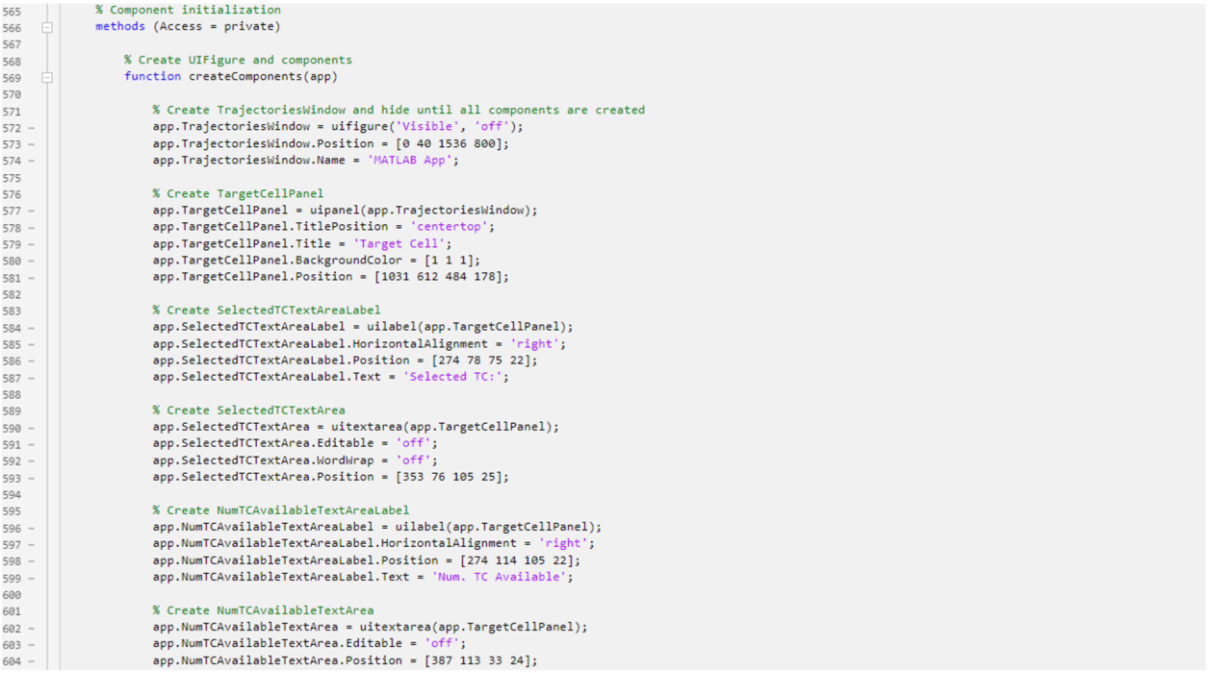


```
                                    app.traj{i}.ReactionTime=e.trials.HoldAFinish(i)+RT;
                                    app.traj{i}.ReactionTimePos=interp1(app.traj{i}.t,app.traj{i}.pos,app.traj{i}.ReactionTime);

                                    if isfield(e.header.cout,'cursorInvisibleZone')
                                        r=e.header.cout.cursorInvisibleZone;
                                        dist=sqrt(sum(app.traj{i}.emfeed_pos.^2,2));
                                        outinds=find(dist>r);
                                        if ~isempty(outinds)
                                            app.traj{i}.InvisExit=app.traj{i}.emfeed_pos(outinds(1),:);
                                            app.traj{i}.InvisExitTime=app.traj{i}.t(outinds(1));
                                        end
                                    end
                                else
                                    discardedtrials(end+1)=i;
                                    app.traj{i}.t=zeros(0,1);
                                    app.traj{i}.firtemtime=zeros(0,1);
                                    app.traj{i}.computerstarttime=zeros(0,1);
                                    app.traj{i}.HoldAStart=zeros(0,1);
                                    app.traj{i}.HoldAFinish=zeros(0,1);
                                    app.traj{i}.HoldBStart=zeros(0,1);
                                    app.traj{i}.HoldBFinish=zeros(0,1);
                                    app.traj{i}.InvisExit=NaN*zeros(0,ndim);
                                    app.traj{i}.InvisExitTime=NaN*zeros(0,1);
                                    app.traj{i}.ReactionTime=NaN*zeros(0,1);
                                    app.traj{i}.ReactionTimePos=NaN*zeros(0,ndim);
                                    app.traj{i}.emcomm_vel=zeros(0,ndim);
                                    app.traj{i}.emcomm_pos=zeros(0,ndim);
                                    app.traj{i}.emfeed_pos=zeros(0,ndim);
                                    app.traj{i}.pos=zeros(0,ndim);
                                end
                                for element = 1:length(app.traj)
                                    hold(ax,"on")
                                    plot(ax, app.traj{element}.pos(:,1),app.traj{element}.pos(:,2))
                                    plot(ax, app.targs(:,1),app.targs(:,2),'ko')
                                    hold(ax, "off")
                                end
                            end
                        end
                    catch err
                        %errordlg(err.message)
                    end
                end
            end

        end
    end
```

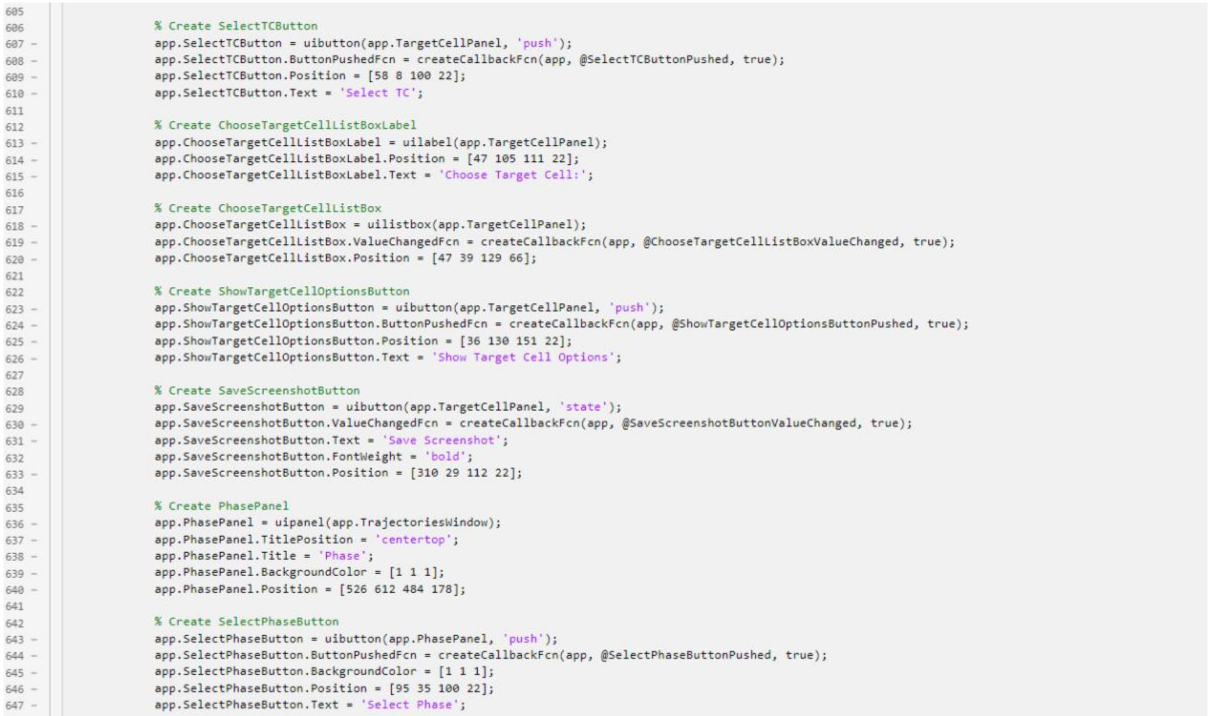


```
        % Value changed function: SaveScreenshotButton
        function SaveScreenshotButtonValueChanged(app, event)

            % Save a screenshot of the window
            filter = {'*.jpg';'*.png';'*.tif';'*.pdf'};         % File type options
            [filename,filepath] = uiputfile(filter);            % Open dialog box for saving files
            if ischar(filename)                                 % If the name has the correct format, save the file
                exportapp(app.TrajectoriesWindow,[filepath filename]);
            end
            app.TrajectoriesWindow.Visible = 'off';               % These two lines of code work-around an issue whether the figure is sent to the background.
            app.TrajectoriesWindow.Visible = 'on';

        end
    end
```

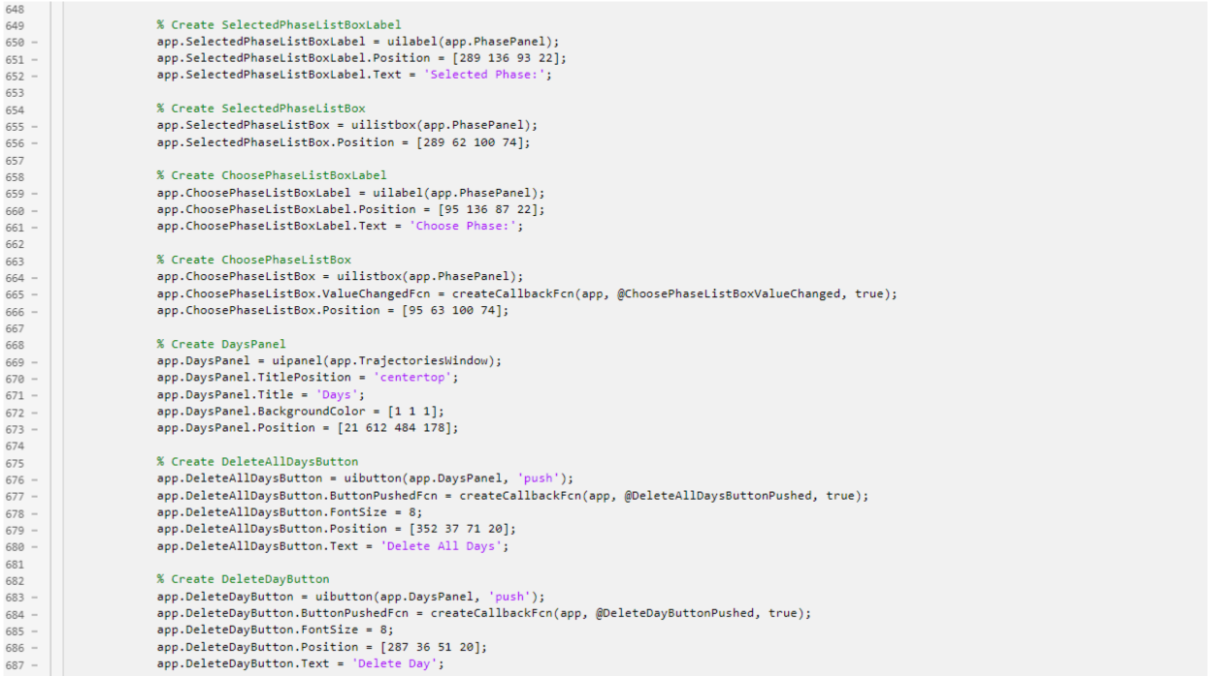
Component initialization



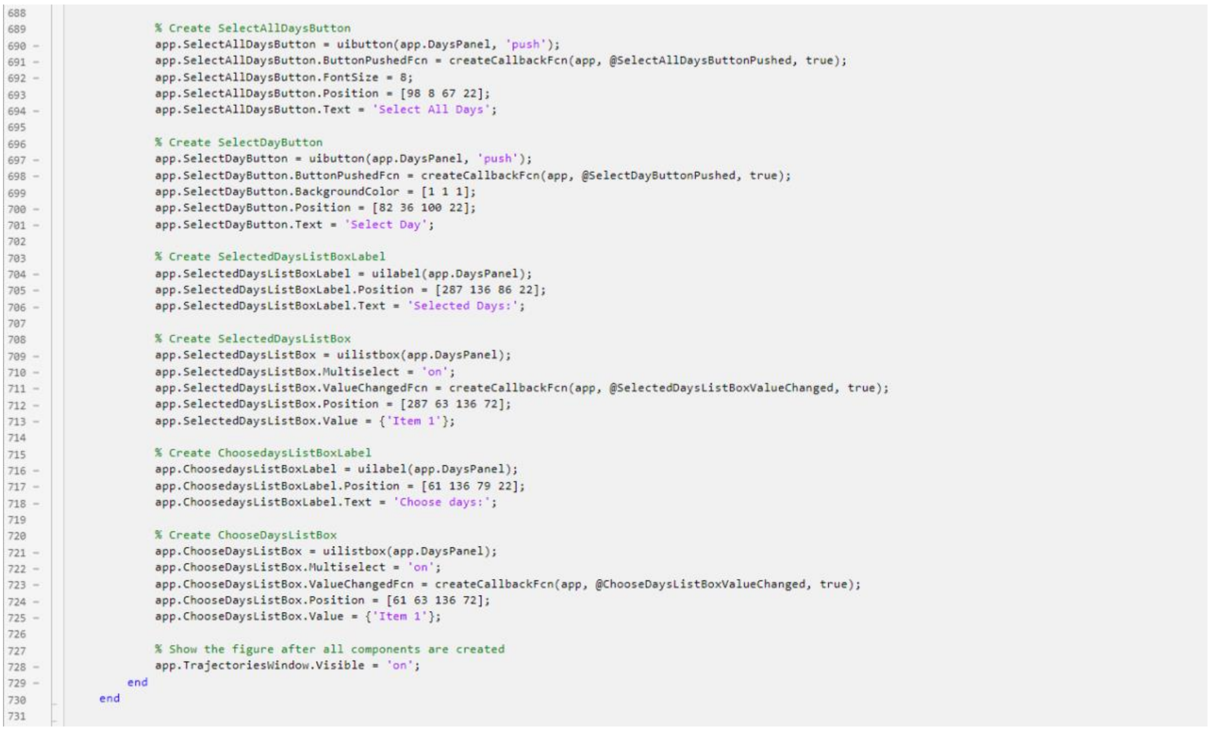
```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create TrajectoriesWindow and hide until all components are created
            app.TrajectoriesWindow = uifigure('Visible', 'off');
            app.TrajectoriesWindow.Position = [0 40 1536 800];
            app.TrajectoriesWindow.Name = 'MATLAB App';

            % Create TargetCellPanel
            app.TargetCellPanel = uipanel(app.TrajectoriesWindow);
            app.TargetCellPanel.TitlePosition = 'centertop';
            app.TargetCellPanel.Title = 'Target Cell';
            app.TargetCellPanel.BackgroundColor = [1 1 1];
            app.TargetCellPanel.Position = [1031 612 484 178];

            % Create SelectedTCTextAreaLabel
            app.SelectedTCTextAreaLabel = uilabel(app.TargetCellPanel);
            app.SelectedTCTextAreaLabel.HorizontalAlignment = 'right';
            app.SelectedTCTextAreaLabel.Position = [274 78 75 22];
            app.SelectedTCTextAreaLabel.Text = 'Selected TC:';

            % Create SelectedTCTextArea
            app.SelectedTCTextArea = uitextarea(app.TargetCellPanel);
            app.SelectedTCTextArea.Editable = 'off';
            app.SelectedTCTextArea.WordWrap = 'off';
            app.SelectedTCTextArea.Position = [353 76 105 25];

            % Create NumTCAvailableTextAreaLabel
            app.NumTCAvailableTextAreaLabel = uilabel(app.TargetCellPanel);
            app.NumTCAvailableTextAreaLabel.HorizontalAlignment = 'right';
            app.NumTCAvailableTextAreaLabel.Position = [274 114 105 22];
            app.NumTCAvailableTextAreaLabel.Text = 'Num. TC Available';

            % Create NumTCAvailableTextArea
            app.NumTCAvailableTextArea = uitextarea(app.TargetCellPanel);
            app.NumTCAvailableTextArea.Editable = 'off';
            app.NumTCAvailableTextArea.Position = [387 113 33 24];
```



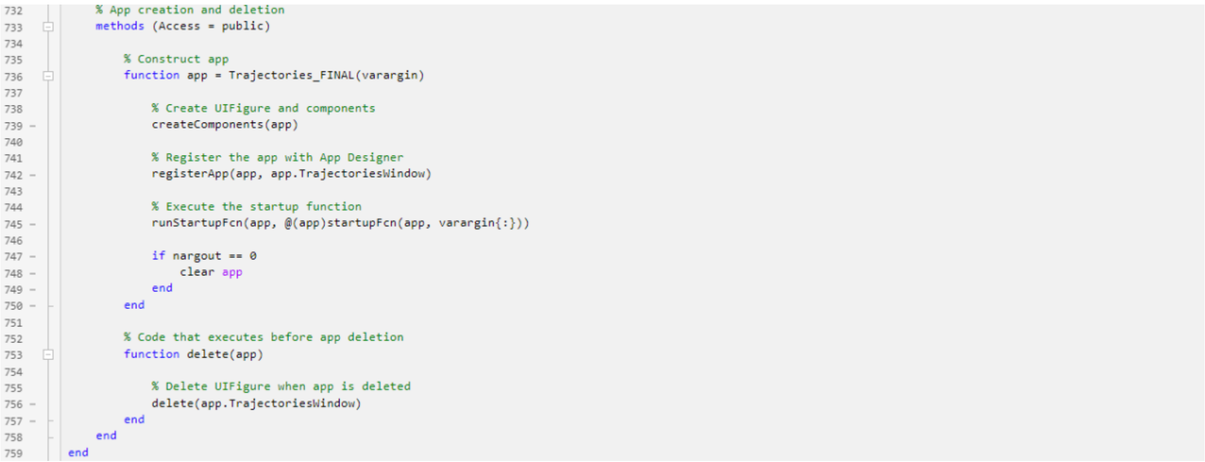
```
            % Create SelectTCButton
            app.SelectTCButton = uibutton(app.TargetCellPanel, 'push');
            app.SelectTCButton.ButtonPushedFcn = createCallbackFcn(app, @SelectTCButtonPushed, true);
            app.SelectTCButton.Position = [58 8 100 22];
            app.SelectTCButton.Text = 'Select TC';

            % Create ChooseTargetCellListBoxLabel
            app.ChooseTargetCellListBoxLabel = uilabel(app.TargetCellPanel);
            app.ChooseTargetCellListBoxLabel.Position = [47 105 111 22];
            app.ChooseTargetCellListBoxLabel.Text = 'Choose Target Cell:';

            % Create ChooseTargetCellListBox
            app.ChooseTargetCellListBox = uilistbox(app.TargetCellPanel);
            app.ChooseTargetCellListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseTargetCellListBoxValueChanged, true);
            app.ChooseTargetCellListBox.Position = [47 39 129 66];

            % Create ShowTargetCellOptionsButton
            app.ShowTargetCellOptionsButton = uibutton(app.TargetCellPanel, 'push');
            app.ShowTargetCellOptionsButton.ButtonPushedFcn = createCallbackFcn(app, @ShowTargetCellOptionsButtonPushed, true);
            app.ShowTargetCellOptionsButton.Position = [36 130 151 22];
            app.ShowTargetCellOptionsButton.Text = 'Show Target Cell Options';

            % Create SaveScreenshotButton
            app.SaveScreenshotButton = uibutton(app.TargetCellPanel, 'state');
            app.SaveScreenshotButton.ValueChangedFcn = createCallbackFcn(app, @SaveScreenshotButtonValueChanged, true);
            app.SaveScreenshotButton.Text = 'Save Screenshot';
            app.SaveScreenshotButton.FontWeight = 'bold';
            app.SaveScreenshotButton.Position = [310 29 112 22];

            % Create PhasePanel
            app.PhasePanel = uipanel(app.TrajectoriesWindow);
            app.PhasePanel.TitlePosition = 'centertop';
            app.PhasePanel.Title = 'Phase';
            app.PhasePanel.BackgroundColor = [1 1 1];
            app.PhasePanel.Position = [526 612 484 178];

            % Create SelectPhaseButton
            app.SelectPhaseButton = uibutton(app.PhasePanel, 'push');
            app.SelectPhaseButton.ButtonPushedFcn = createCallbackFcn(app, @SelectPhaseButtonPushed, true);
            app.SelectPhaseButton.BackgroundColor = [1 1 1];
            app.SelectPhaseButton.Position = [95 35 100 22];
            app.SelectPhaseButton.Text = 'Select Phase';
```



```
            % Create SelectedPhaseListBoxLabel
            app.SelectedPhaseListBoxLabel = uilabel(app.PhasePanel);
            app.SelectedPhaseListBoxLabel.Position = [289 136 93 22];
            app.SelectedPhaseListBoxLabel.Text = 'Selected Phase:';

            % Create SelectedPhaseListBox
            app.SelectedPhaseListBox = uilistbox(app.PhasePanel);
            app.SelectedPhaseListBox.Position = [289 62 100 74];

            % Create ChoosePhaseListBoxLabel
            app.ChoosePhaseListBoxLabel = uilabel(app.PhasePanel);
            app.ChoosePhaseListBoxLabel.Position = [95 136 87 22];
            app.ChoosePhaseListBoxLabel.Text = 'Choose Phase:';

            % Create ChoosePhaseListBox
            app.ChoosePhaseListBox = uilistbox(app.PhasePanel);
            app.ChoosePhaseListBox.ValueChangedFcn = createCallbackFcn(app, @ChoosePhaseListBoxValueChanged, true);
            app.ChoosePhaseListBox.Position = [95 63 100 74];

            % Create DaysPanel
            app.DaysPanel = uipanel(app.TrajectoriesWindow);
            app.DaysPanel.TitlePosition = 'centertop';
            app.DaysPanel.Title = 'Days';
            app.DaysPanel.BackgroundColor = [1 1 1];
            app.DaysPanel.Position = [21 612 484 178];

            % Create DeleteAllDaysButton
            app.DeleteAllDaysButton = uibutton(app.DaysPanel, 'push');
            app.DeleteAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteAllDaysButtonPushed, true);
            app.DeleteAllDaysButton.FontSize = 8;
            app.DeleteAllDaysButton.Position = [352 37 71 20];
            app.DeleteAllDaysButton.Text = 'Delete All Days';

            % Create DeleteDayButton
            app.DeleteDayButton = uibutton(app.DaysPanel, 'push');
            app.DeleteDayButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteDayButtonPushed, true);
            app.DeleteDayButton.FontSize = 8;
            app.DeleteDayButton.Position = [287 36 51 20];
            app.DeleteDayButton.Text = 'Delete Day';
```



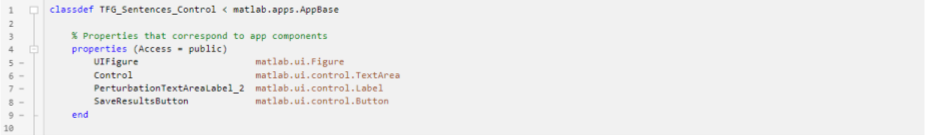
```

            % Create SelectAllDaysButton
            app.SelectAllDaysButton = uibutton(app.DaysPanel, 'push');
            app.SelectAllDaysButton.ButtonPushedFcn = createCallbackFcn(app, @SelectAllDaysButtonPushed, true);
            app.SelectAllDaysButton.FontSize = 8;
            app.SelectAllDaysButton.Position = [98 8 67 22];
            app.SelectAllDaysButton.Text = 'Select All Days';

            % Create SelectDayButton
            app.SelectDayButton = uibutton(app.DaysPanel, 'push');
            app.SelectDayButton.ButtonPushedFcn = createCallbackFcn(app, @SelectDayButtonPushed, true);
            app.SelectDayButton.BackgroundColor = [1 1 1];
            app.SelectDayButton.Position = [82 36 100 22];
            app.SelectDayButton.Text = 'Select Day';

            % Create SelectedDaysListBoxLabel
            app.SelectedDaysListBoxLabel = uilabel(app.DaysPanel);
            app.SelectedDaysListBoxLabel.Position = [287 136 86 22];
            app.SelectedDaysListBoxLabel.Text = 'Selected Days:';

            % Create SelectedDaysListBox
            app.SelectedDaysListBox = uilistbox(app.DaysPanel);
            app.SelectedDaysListBox.Multiselect = 'on';
            app.SelectedDaysListBox.ValueChangedFcn = createCallbackFcn(app, @SelectedDaysListBoxValueChanged, true);
            app.SelectedDaysListBox.Position = [287 63 136 72];
            app.SelectedDaysListBox.Value = {'Item 1'};

            % Create ChoosedaysListBoxLabel
            app.ChoosedaysListBoxLabel = uilabel(app.DaysPanel);
            app.ChoosedaysListBoxLabel.Position = [61 136 79 22];
            app.ChoosedaysListBoxLabel.Text = 'Choose days:';

            % Create ChooseDaysListBox
            app.ChooseDaysListBox = uilistbox(app.DaysPanel);
            app.ChooseDaysListBox.Multiselect = 'on';
            app.ChooseDaysListBox.ValueChangedFcn = createCallbackFcn(app, @ChooseDaysListBoxValueChanged, true);
            app.ChooseDaysListBox.Position = [61 63 136 72];
            app.ChooseDaysListBox.Value = {'Item 1'};

            % Show the figure after all components are created
            app.TrajectoriesWindow.Visible = 'on';
        end
    end

```



```
    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Trajectories_FINAL(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.TrajectoriesWindow)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.TrajectoriesWindow)
        end
    end
end
```

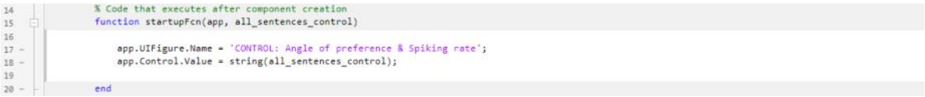
ANNEX IV. Code used for the development of the Control results window

Properties



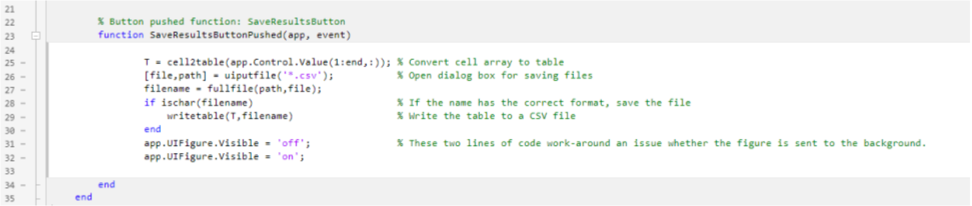
```
classdef TFG_Sentences_Control < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                     matlab.ui.Figure
        Control                      matlab.ui.control.TextArea
        PerturbationTextAreaLabel_2  matlab.ui.control.Label
        SaveResultsButton            matlab.ui.control.Button
    end

```

Startup function



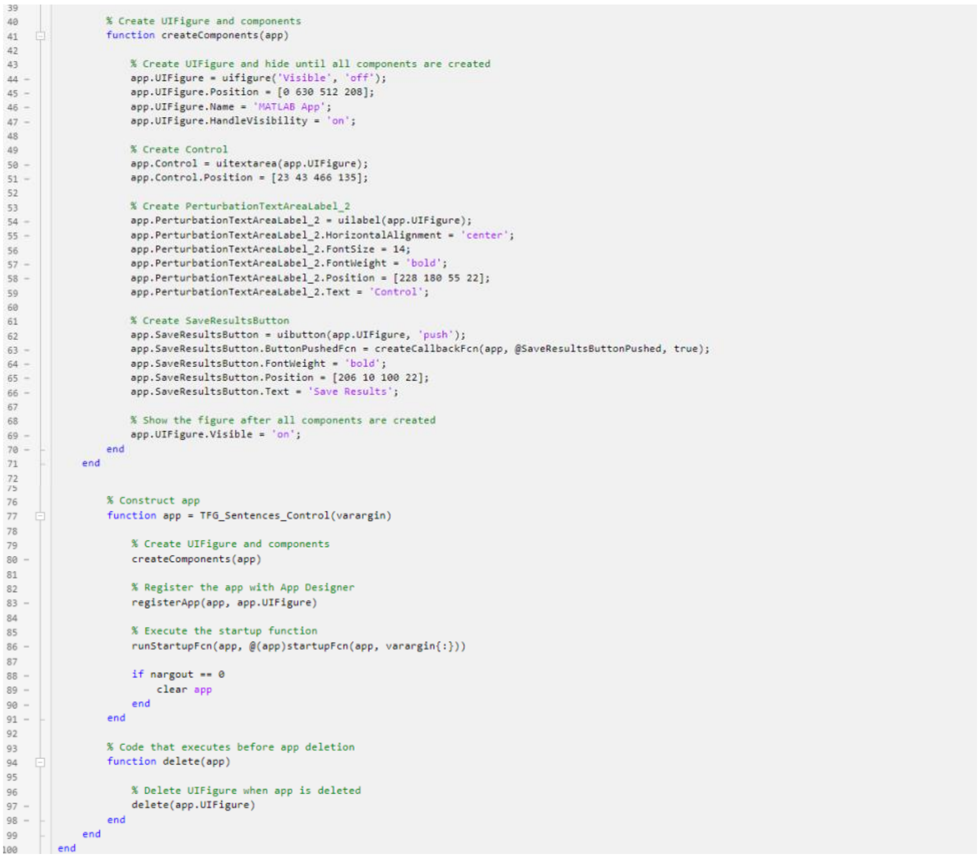
```
        % Code that executes after component creation
        function startupFcn(app, all_sentences_control)

            app.UIFigure.Name = 'CONTROL: Angle of preference & Spiking rate';
            app.Control.Value = string(all_sentences_control);

        end
```

Callback functions



```

        % Button pushed function: SaveResultsButton
        function SaveResultsButtonPushed(app, event)

            T = cell2table(app.Control.Value(1:end,:)); % Convert cell array to table
            [file,path] = uiputfile('*.csv');           % Open dialog box for saving files
            filename = fullfile(path,file);
            if ischar(filename)                         % If the name has the correct format, save the file
                writetable(T,filename)                  % Write the table to a CSV file
            end
            app.UIFigure.Visible = 'off';               % These two lines of code work-around an issue whether the figure is sent to the background.
            app.UIFigure.Visible = 'on';

        end
    end
```

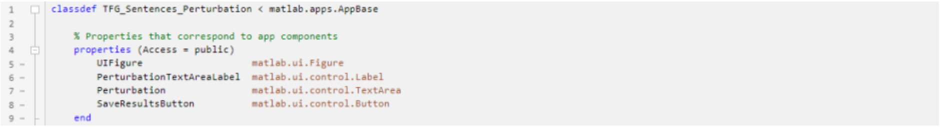
Component initialization



```

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [0 630 512 208];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.HandleVisibility = 'on';

            % Create Control
            app.Control = uitextarea(app.UIFigure);
            app.Control.Position = [23 43 466 135];

            % Create PerturbationTextAreaLabel_2
            app.PerturbationTextAreaLabel_2 = uilabel(app.UIFigure);
            app.PerturbationTextAreaLabel_2.HorizontalAlignment = 'center';
            app.PerturbationTextAreaLabel_2.FontSize = 14;
            app.PerturbationTextAreaLabel_2.FontWeight = 'bold';
            app.PerturbationTextAreaLabel_2.Position = [228 180 55 22];
            app.PerturbationTextAreaLabel_2.Text = 'Control';

            % Create SaveResultsButton
            app.SaveResultsButton = uibutton(app.UIFigure, 'push');
            app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
            app.SaveResultsButton.FontWeight = 'bold';
            app.SaveResultsButton.Position = [206 10 100 22];
            app.SaveResultsButton.Text = 'Save Results';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end


        % Construct app
        function app = TFG_Sentences_Control(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

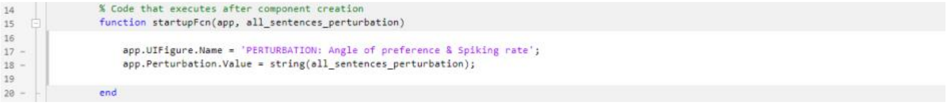
ANNEX V. Code used for the development of the Perturbation results window

Properties



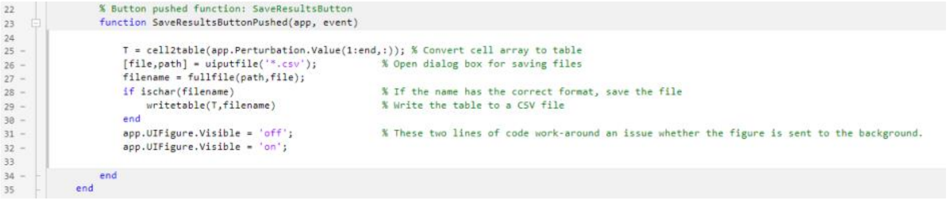
```matlab
classdef TFG_Sentences_Perturbation < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        PerturbationTextAreaLabel   matlab.ui.control.Label
        Perturbation                matlab.ui.control.TextArea
        SaveResultsButton           matlab.ui.control.Button
    end
```

Startup function



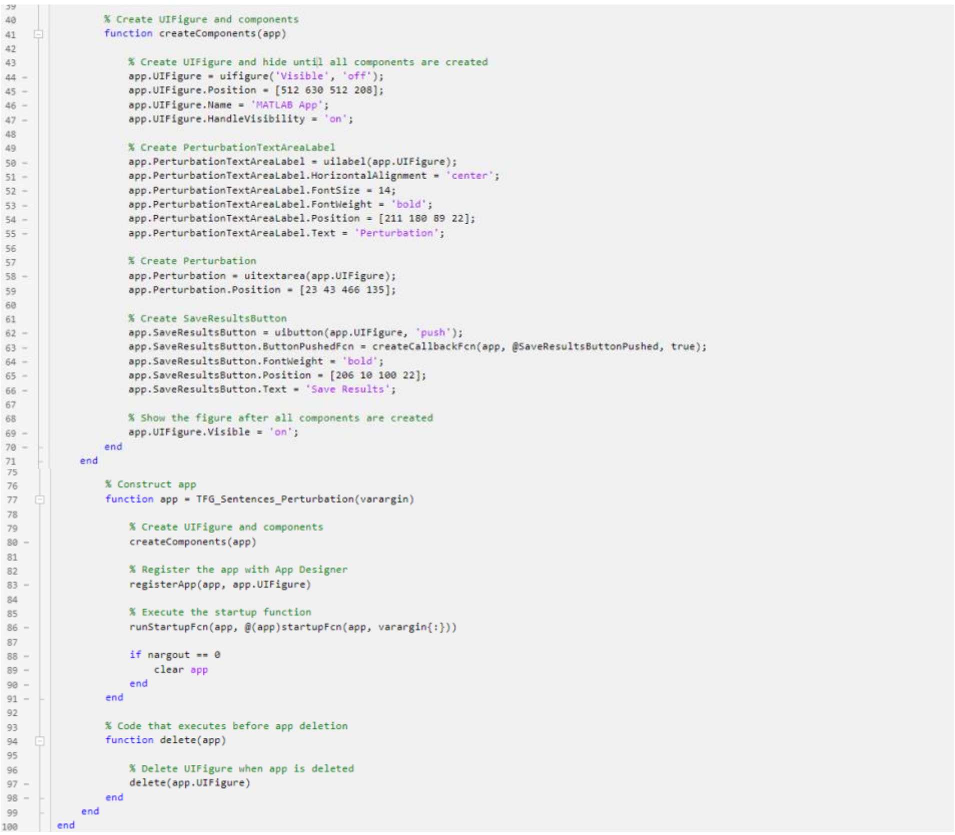
```matlab
        % Code that executes after component creation
        function startupFcn(app, all_sentences_perturbation)

            app.UIFigure.Name = 'PERTURBATION: Angle of preference & Spiking rate';
            app.Perturbation.Value = string(all_sentences_perturbation);

        end
```

Callback functions



```matlab
        % Button pushed function: SaveResultsButton
        function SaveResultsButtonPushed(app, event)

            T = cell2table(app.Perturbation.Value(1:end,:)); % Convert cell array to table
            [file,path] = uiputfile('*.csv');            % Open dialog box for saving files
            filename = fullfile(path,file);
            if ischar(filename)                          % If the name has the correct format, save the file
                writetable(T,filename)                   % Write the table to a CSV file
            end
            app.UIFigure.Visible = 'off';                % These two lines of code work-around an issue whether the figure is sent to the background.
            app.UIFigure.Visible = 'on';

        end
    end
```

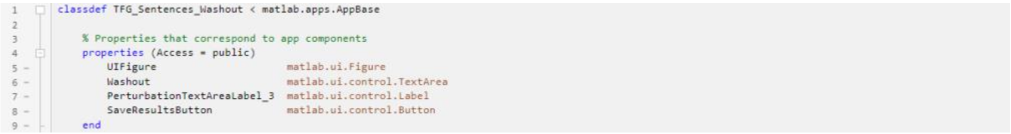
Component initialization



```matlab
        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [512 630 512 208];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.HandleVisibility = 'on';

            % Create PerturbationTextAreaLabel
            app.PerturbationTextAreaLabel = uilabel(app.UIFigure);
            app.PerturbationTextAreaLabel.HorizontalAlignment = 'center';
            app.PerturbationTextAreaLabel.FontSize = 14;
            app.PerturbationTextAreaLabel.FontWeight = 'bold';
            app.PerturbationTextAreaLabel.Position = [211 180 89 22];
            app.PerturbationTextAreaLabel.Text = 'Perturbation';

            % Create Perturbation
            app.Perturbation = uitextarea(app.UIFigure);
            app.Perturbation.Position = [23 43 466 135];

            % Create SaveResultsButton
            app.SaveResultsButton = uibutton(app.UIFigure, 'push');
            app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
            app.SaveResultsButton.FontWeight = 'bold';
            app.SaveResultsButton.Position = [206 10 100 22];
            app.SaveResultsButton.Text = 'Save Results';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

        % Construct app
        function app = TFG_Sentences_Perturbation(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

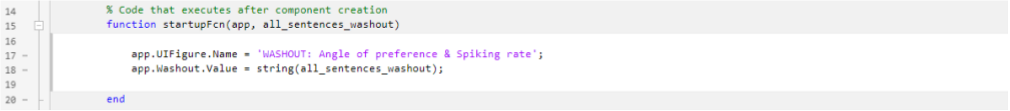
ANNEX VI. Code used for the development of the Washout results window

Properties



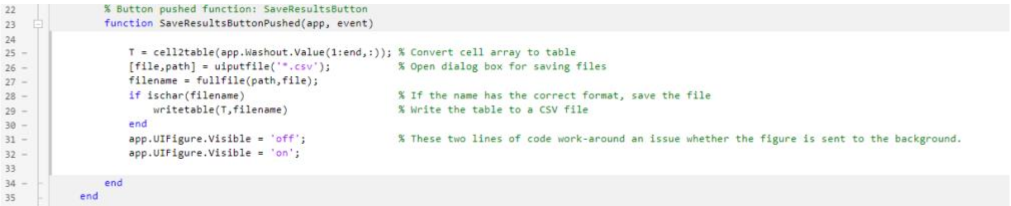
```
classdef TFG_Sentences_Washout < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        Washout                     matlab.ui.control.TextArea
        PerturbationTextAreaLabel_3 matlab.ui.control.Label
        SaveResultsButton           matlab.ui.control.Button
    end
```

Startup function



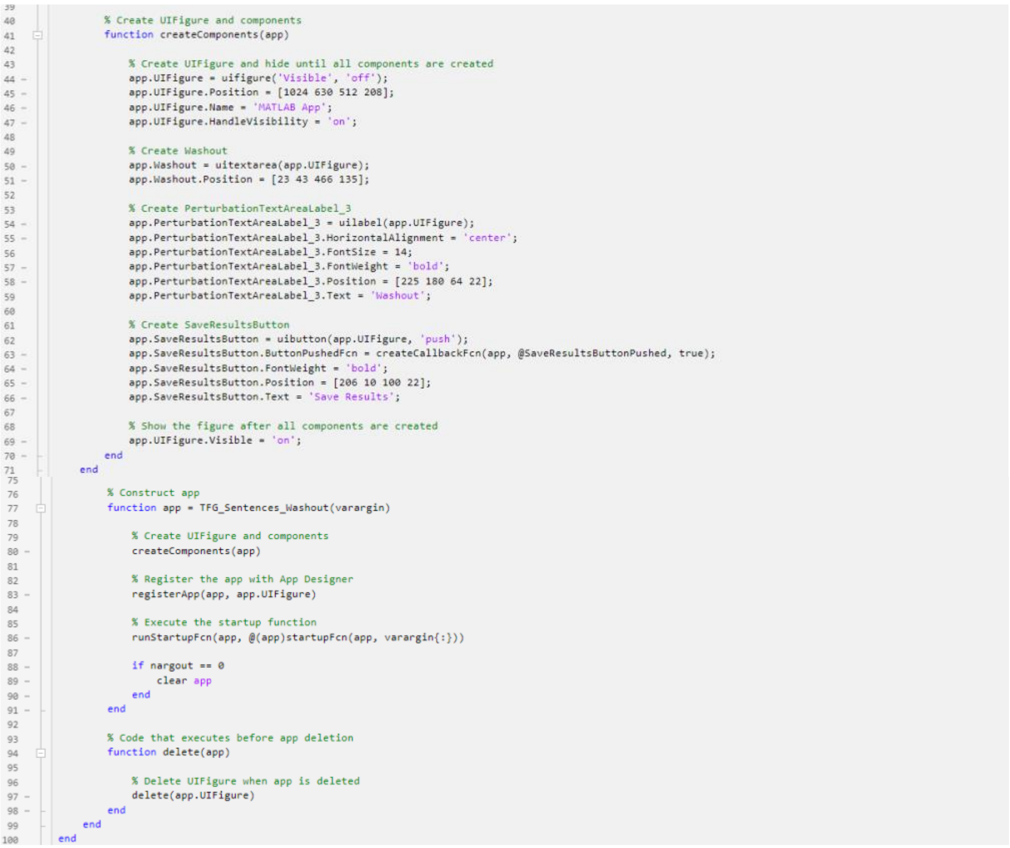
```
        % Code that executes after component creation
        function startupFcn(app, all_sentences_washout)

            app.UIFigure.Name = 'WASHOUT: Angle of preference & Spiking rate';
            app.Washout.Value = string(all_sentences_washout);

        end
```

Callback functions



```
        % Button pushed function: SaveResultsButton
        function SaveResultsButtonPushed(app, event)

            T = cell2table(app.Washout.Value(1:end,:)); % Convert cell array to table
            [file,path] = uiputfile('*.csv');           % Open dialog box for saving files
            filename = fullfile(path,file);
            if ischar(filename)                         % If the name has the correct format, save the file
                writetable(T,filename)                  % Write the table to a CSV file
            end
            app.UIFigure.Visible = 'off';               % These two lines of code work-around an issue whether the figure is sent to the background.
            app.UIFigure.Visible = 'on';

        end
    end
```

Component initialization



```
        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [1024 630 512 208];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.HandleVisibility = 'on';

            % Create Washout
            app.Washout = uitextarea(app.UIFigure);
            app.Washout.Position = [23 43 466 135];

            % Create PerturbationTextAreaLabel_3
            app.PerturbationTextAreaLabel_3 = uilabel(app.UIFigure);
            app.PerturbationTextAreaLabel_3.HorizontalAlignment = 'center';
            app.PerturbationTextAreaLabel_3.FontSize = 14;
            app.PerturbationTextAreaLabel_3.FontWeight = 'bold';
            app.PerturbationTextAreaLabel_3.Position = [225 180 64 22];
            app.PerturbationTextAreaLabel_3.Text = 'Washout';

            % Create SaveResultsButton
            app.SaveResultsButton = uibutton(app.UIFigure, 'push');
            app.SaveResultsButton.ButtonPushedFcn = createCallbackFcn(app, @SaveResultsButtonPushed, true);
            app.SaveResultsButton.FontWeight = 'bold';
            app.SaveResultsButton.Position = [206 10 100 22];
            app.SaveResultsButton.Text = 'Save Results';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

        % Construct app
        function app = TFG_Sentences_Washout(varargin)

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @(app)startupFcn(app, varargin{:}))

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

ANNEX VII. List of Figures and Tables

List of figures

List of tables